TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät

Insinöörityö

Jani Angervuo

HITAUSMOMENTIN SIMULOINTILAITTEISTON OHJAUS

Työn valvoja: Lehtori, Ilkka Tervaoja
Työn ohjaaja: Insinööri, Jukka Korpela
Tampere 2007

Tekijä: Jani Angervuo
Työn nimi: HITAUSMOMENTIN SIMULOINTILAITTEISTON OHJAUS

Päivämäärä: 06.03.2007
Sivumäärä: 41 sivua + 18 liitesivua
Hakusanat: hitausmomentti, simulointi, ohjainkortti
Koulutusohjelma: Tietotekniikka
Suuntautumisvaihtoehto: Tietokonetekniikka / Sulautetut järjestelmät

Työn valvoja: Ilkka Tervaoja, lehtori

Työn ohjaaja: Jukka Korpela, Reliability Manager, insinööri

Elektroniikan testaus on nykyisin tärkeässä roolissa niin tuotanto- kuin tuotekehitysprosesseissa. Kokoonpanolaitteiden edelleen nopeutuessa ja komponenttien pienentyessä virheiden määrä lisääntyy, vaikka kokoonpanolaitteiden tarkkuus paranee myös samalla, kun laitteita kehitetään yhä tuottavammiksi.

Tuotantoprosessin lopuksi valmistettu laite testataan yleisesti, koska tällaiset testaus-toiminteet kuuluvat yleisesti yritysten laatujärjestelmiin. Testauksen laatu riippuu yleensä tuotteen tilaajasta, tarvitaanko tai halutaanko tuotantotestausta tehdä.
Laitetestaus tapahtuu yleensä niin sanotulla In-Circuit–testauksella, mutta myös toiminnallinen testaus on nykypäivänä melko normaalia.

Tässä tutkintotyössä tutkittiin laitteen toteuttamismahdollisuudet ja rakennettiin toiminnallisen testauksen tueksi ja helpottamiseksi testilaitteen ohjauskortti. Koko projektin tavoitteena on rakentaa toiminnallinen ympäristötestauslaite hissin ovioperaattorin testaukseen. Ovioperaattorikortti ohjaa moottorin avulla hissin ovea, jonka mekaanisten rakenteiden aiheuttaman massan hitausmomentti- rasituksen simulointi on laitteen päätarkoitus. Laite simuloi myös ovioperaattorin ohjauksessa olevia mekaanisten turvalaitteiden sähköisiä signaaleja.

Poikittaisliikkeisen massan aiheuttaman hitausmomentin simulointi yhdessä mekaanisten rajojen eli oven auki- ja kiinni-rajojen kanssa on melko vaativaa, kun käytetään simulointiin toista sähkömoottoria ilman mekaanista jarrua. Käytössä on siis vain kaksi vapaasti pyörivää roottoria, jotka on yhdistetty toisiinsa.

Kuormitusmoottorin ohjauksessa tulee kyetä ottamaan huomioon nopeat muutokset, ja rajojen simuloinnissa täytyy moottori saada pysähtymään nopeasti ja hallitusti, vaikka roottorin akselilla vaikuttaa edelleen sama ovimoottorin aiheuttama vääntömomentti.

Työ koskettaa kokonaisen laitteiston ohjauskorttia, jolla järjestelmää hallitaan. Työssä edetään vaiheittain laitteen määrittelystä ja ohjainkortin vaatimusten määrittelystä ohjainkortin suunnitteluun ja sen toteutukseen.

Suunnittelu ja toteutus kuvataan työssä yhdessä, jotta toiminnalliset ohjainkortin kokonaisuudet on saatu kuvattua lukijalle helpommin ymmärrettävässä muodossa. Elektronisen suunnittelun ja toteutuksen jälkeen käsitellään myös ohjausohjelmisto, jonka avulla ohjaus tapahtuu.

Kokonaisen laitteen toiminnallinen selostus ja sen muu tekninen dokumentointi jää KONE Oyj:n käyttöön.

Author: Jani Angervuo
Title: CONTROLLER FOR INERTIA SIMULATIONMACHINE

Date: 06.03.2007
Number of pages: 41 pages + 18 attachment pages
Keywords: Moment of inertia, simulation, controller board
Program: Computer Systems Engineering
Specialization: Computer technology / Embedded systems

Supervisor: Ilkka Tervaoja, lecturer

Instructor: Jukka Korpela, Reliability Manager, professional engineer

Testing of electronics has a big role on production and research processes today. Electronics assembly devices are coming more and more rapid and component size is decreasing all the time. This will cause more failures on production process even assembly devices accuracy is growing also at the same time.

When electronic board is assembled, it is usually tested by factory, because of the quality systems of companies. Quality of testing is usually depend of a subscriber of the device. Subscriber defines need of testing of electronic boards.
Usually testing of circuit board is made with In-Circuit testing. Today also functional testing is used more and more beside ICT.

Target of this thesis project is a check possibilities and make controller board for testing system for functional testing of elevator door operator boards. Whole project target is build whole functional testing environment. Door operator board is controlling elevator door by a type-defined motor. Testing system is simulating mechanical stress to door motor, which is caused by elevator door mechanically moving parts. There need to be also some electric signals, which are simulated also by tester system.

Torque of inertia by mass in vertical movement is quite complicated when there needs to be also simulation of mechanical limits of door, when using electric motor without mechanical break. There are only two free rotating motor axels to fixed together.
Simulator motor control is important that motor torque can be changed fast and simulating mechanical limits motor is need to be stopped fast and well controlled, even there is still same torque on axel from door motor.

This thesis includes only a controller board of whole system. The workflow proceeds from defining the functional specification to designing and implementing the controller board and finally to complete testing of controller board.

Designing and implementation of board are handled together to give better view to reader about functional integrities. After electronics design and implementing there is also explanation about software design and its functional description. Software is controlling whole controller board.

Full system descriptions and other technical documents are owned by KONE Corporation and will be published only in internal use.

ALKUSANAT

Kiitokset KONE Oyj:n luotettavuuslaboratoriolle mahdollisuudesta tehdä tämä tutkintotyö siten kuin sen tarkoitus on, näytteeksi opintojen suorittamisesta, elinkeinoelämän tarpeiden mukaisesti.

Erityinen kiitos siitä, että tutkintotyön kartoitukseen ja tekemiseen käytetty aika oli joustavaa ja näin ollen työ oli hyvin sopiva opintojen ohessa tehtäväksi.

Kiitokset kuluneesta opiskelu-ajasta Tampereen ammattikorkeakoulun hallinto- ja koulutushenkilöstölle. Teidän halunne kuunnella opiskelijoita ja kehittää ammattikorkeakoulun toimintaa ja opetusta on ollut hienoa seurattavaa. Kiitokset myös kehittävistä keskusteluista ja kehittämisessä mukana olon mahdollisuudesta.

Lisäksi erityinen kiitos Urho Honkaselle, Arto Nakarille sekä Päivi ja Kari Angervuolle, jotka ovat olleet mukana tämän tutkintotyön sisällön ja kirjoitusasun tarkastuksessa työn ohjaaja, Jukka Korpelan lisäksi.

Tampereella 6.3.2007

Jani Angervuo

KÄYTETYT TERMIT JA LYHENTEET

Ovioperaattori	Hissin ovea ohjaava elektroninen ohjainkortti
Ovimoottori	Hissin ovessa käytetty DC-moottori, joka on ovioperaattorityypin mukaan joko tavallinen DC-moottori tai kolmivaiheinen harjaton DC-moottori
Kuormittava moottori	Simulointilaitteiston moottori jolla luodaan massan hitausmomenttia vastaava vääntömomentti
AMD1 ja AMD2	Ovioperaattorityyppi
Oviprofiili	Oven mekaaniset mitat, joiden mukaan simulointi tapahtuu
ICT	In Circuit Testing – testaustapa tai laitteisto, jonka avulla piirilevyn toiminta voidaan testata ohmisesti jokaisen komponentin kohdalta.
FCT	Functional Circuit Testing – testaustapa jonka avulla voidaan testata laitteen toimivuus syötteen ja tuloksen perusteella
AVR	Atmel Corporationin valmistama 8-bittinen mikro-ohjainperhe
ATmega	AVR-tuoteperheen nykyisin tuotannossa oleva tuotesarja
CMOS	Complementary Metal-Oxide Semiconductor – pienen tehonkulutuksen omaava puolijohdetekniikka
RISC	Reduced Instruction Set Computer, rajoitetun käskykannan mikroprosessori, jonka käskykannan käskyt suoritetaan yhden kellojakson aikana
Flash	Muisteissa käytetty puolijohdetekniikka, uudelleen ohjelmoitava
EEPROM	Muisteissa käytetty puolijohdetekniikka, uudelleen ohjelmoitava, hitaampi kuin Flash-muisti
SRAM	Static Random Access Memory, nopea käyttömuisti
UART	Universal Asynchronous Receiver/Transmitter – asynkroninen sarjaliikennepiiri
A/D-muunnin	Analog-to-Digital-muunnin, muuntaa analogisen jännitteen digitaaliseksi sanaksi
D/A-muunnin	Digital-to-Analog-muunnin, muuntaa digitaalisen sanan analogiseksi jännitteeksi
RS-232	Yleisesti käytetty sarjaliikenneväylän standardi
C-kieli	Korkean tason ohjelmointikieli, joka voidaan kääntää konekielelle
ISO C99	C-kielen standardi vuodelta 1999
WinAVR	GNU GPL-lisenssin alainen C-kääntäjäympäristö
CodeVision AVR	Kaupallinen C-kääntäjäympäristö AVR-tuoteperheen ohjaimille
LCD	Liquid Crystal Display - nestekidenäyttö
I/O	Input/Output, digitaalisissa piireissä käytetty lyhenne sisääntulo- ja lähtöporteista
GNU GPL	GNU General Public License, vapaaseen käyttöön tarkoitetun ohjelmiston lisenssi
Makefile	WinAVR:n (ja GNU CC -kääntäjän) aputiedosto jossa määritellään miten ohjelma käännetään C-kielestä assembler-kielen kautta konekieliseksi ohjelmaksi

SISÄLLYSLUETTELO

1 JOHDANTO

Tämä opinnäytetyö on tehty KONE Oyj:lle ja KONE Oyj:n tarpeiden mukaan syksyn 2006 ja kevään 2007 aikana.

Nykyisin elektroniikkakomponenttien sekä elektronisten laitteiden kokoonpano- ja valmistusprosesseissa kiinnitetään huomiota tuottavuuteen, nopeaan valmistukseen ja laitteiden nopeaan saatavuuteen. Nopeuden lisääntyessä myös laitteiden viat, jotka johtuvat valmistusmenetelmistä, kuten komponenttien puuttumisesta, viallisista komponenteista, juotosvioista ja jopa vääristä komponenteista, lisääntyvät. Tällaiset viat ovat yleisiä varsinkin silloin kun tuote on uusi tai sitä muutetaan. Laitteiden laatuun on syytä kiinnittää huomiota varsinkin laitteen tilaajan taholta ja siksi onkin kehitetty mitä moninaisimpia tapoja testata laitteistoa heti kun se on valmistettu tehtaalla, ennen kuin se toimitetaan lopputilaajalle.

Testaustapoja on kehitetty ja kehitetään koko ajan entistä paremmiksi ja paremmin kattaviksi. Laitteet testataan muun muassa visuaalisesti konenäkölaitteilla tai pienien valmistuserien, kuten prototyyppi-vaiheessa olevien laitteiden osalta ihmissilmän avulla, lisäksi sähköisesti johtavuusmittauksin (niin sanottu ICT-testaus) sekä toiminnallisesti, jolloin koko laitteiston toimintaa testataan erillisellä laitteella, joka simuloi oikeaa ympäristöä (FCT-testaus). Tällaista simulointia voidaan käyttää esimerkiksi mikroprosessorikorteille antaen kortille erilaisia käskyjä ja testaamalla kyseisen käskyn tapahtumat ulkoisesti. Tällaiset testausmenetelmät tuovat laitteelle lisäkustannuksia, mutta parantavat laitteen laatua loppukäyttäjän näkökulmasta. Niiden tarve ja toteutus on lähes aina verrannollinen tuotantomäärään ja kustannuksiin. Mitä suurempi on tuotteen menekki, sitä edullisempaa laitteen testaus on suhteessa laitteen tuottamaan tulokseen, samalla kuitenkin myös testauskustannuksen viemän ajan kustannukset lisääntyvät.

Nyt rakennettava laite toimii FCT-testauksessa, jolla simuloidaan sähköisten signaalien lisäksi myös oikean ympäristön eli hissin oven mekaanisten komponenttien aiheuttamaa rasitusta ovimoottorille ja sitä ohjaavalle ovioperaattorille. Oven mekaanisten komponenttien aiheuttama kuormitus moottorille ja operaattorille perustuu massan hitausmomenttiin. Kuormitus-simulaattoria voidaan käyttää tuotantotestaukseen, jolloin tuotantotestaus toimii niin todenmukaisessa ympäristössä kuin mahdollista ilman että varsinaista, kokonaista ovikoneistoa tarvitsee rakentaa jokaiselle testauspaikalle. Lisäksi simulaattoria voidaan käyttää tuotekehityksen tukena vianhaussa ja toimintatestauksessa.

Tutkintotyön idea, kuormitussimulaattori hissin ovi-operaattorille, lähti laboratorion tarpeista ja hyvistä kokemuksista kyseisenlaisella simulointitavalla, jota käytetään jo esimerkiksi simuloimaan hissin korin ja kuilun aiheuttamia rasituksia hissikoneistolle.

Tutkintotyössä kuvataan testauslaitteen ohjauskortin suunnittelu, toteutus ja sen toiminta.

2 TOIMINNALLINEN MÄÄRITTELY

2.1 Toimintaperiaate ja työn tavoite

Laitteen tulee simuloida alkuperäisen hissin oven mekaanisen toteutuksen tuottamaa kuormaa ovikoneiston moottorille mahdollisimman todenmukaisesti toisen sähkömoottorin avulla. Vastamoottori kytketään suoraan ovimoottorin akselille.

Laite simuloi ovikuormia AMD1 ja AMD2 ovioperaattoreille, jolloin mahdollisten oviprofiilien määrän tulee vastata reaalisten standardiovien määrää. Laitteelle tulee voida syöttää myös ovityyppi parametrein.

2.2 Laitteiston vaatimukset

Laite on yhteensopiva molempien ovityyppien kanssa.

Kuormittavaa moottoria tulee ohjata kaupallisella taajuusmuuttajalla.

Tutkintotyössä kartoitetaan, voidaanko laite integroida alkuperäiseen mekaniikkaan, jonka voimansiirtoon käytetään hihnavetoa, vai rakennetaanko kompaktimpi laitteisto jossa moottorien akselit yhdistetään.

Laitteistoa tulee ohjata erillisellä ohjauskortilla.

2.3 Toiminnalliset vaatimukset

Laitteisto toimii itsenäisesti ja se voidaan parametroida itsenäisesti.

Laitteisto voidaan kytkeä tietokoneeseen parametrointia ja tiedon tallennusta varten.

Laitteisto mittaa itsenäisesti käytettävän virran ja jännitteen arvot joka ajokerralla.

Tulokset voidaan tallentaa tarvittavasta määrästä ajoja laitteen muistiin.

Tulokset ovat suoraan luettavissa sarjaporttiliitännästä tietokoneelle.

Laitteisto sisältää standardioviprofiileja sekä täysin ohjelmoitavan oviprofiilin.

Laitteisto simuloi ovikuorman kulloinkin käytettävän profiilin mukaan.

Laitteistolla voidaan simuloida erilaisia vikatilanteita, joita hissin ovissa ilmenee.

3 LAITTEISTON SUUNNITTELU YLEISESTI

Laitteiston toiminnallisen määrittelyn mukaisesti kartoitettiin ensin moottorinohjainyksikön eli taajuusmuuttajan tarpeet, ja sen jälkeen valittiin käytettävä taajuusmuuttaja, joka sopii ominaisuuksiltaan laitteistoon.

Jotta taajuusmuuttajan ominaisuuksia ja tehon tarvetta voitiin kartoittaa, piti kuitenkin ensin mitoittaa vastavoiman tuottava moottori.

Testattava kokonaisuus eli hissin oven ohjaus sisältää ovioperaattorin ja ovimoottorin, jonka tyyppi on riippuvainen ovioperaattorin tyypistä. Näin ollen moottorin mitoitukseen täytyi käyttää vahvemman ovioperaattorin käyttämää moottoria. Tämän moottorin maksimimomentti sen akselilla on noin 260 Ncm ja koneiston alkuperäisen vetopyörän halkaisijan ulkoreunalla 2,6Nm.

Laitteiston rakenteessa päädyttiin omaan mekaaniseen toteutukseen, sillä ovikoneiston voimansiirto tapahtuu alkuperäisessä mekaniikassa hihnavetoisesti. Jos laitteessa olisi käytetty hihnavoimansiirtoa, olisi mitä suurimmalla todennäköisyydellä törmätty hihnan liialliseen luistamiseen ja venymiseen kahden eri suuntaan pyörivän moottorin takia. Tällöin hihnan luisto olisi tuottanut ohjaukseen vaikeuksia ja hyvinkin epäluotettavan vastavoiman simuloinnin.

Näin ollen rakennettiin oma mekaaninen toteutus, jossa testattavan laitteen ja testilaitteen moottorit kytkettiin yhteen suoraan akseleistaan. Moottoreiden akseleille ei kytketty mekaanista jarrua, vaan vastavoimamoottorin tulee kyetä pysäyttämään liike ja näin simuloimaan oven auki- ja kiinni-rajoja.

Nyt voitiin sanoa, että valittavan moottorin tuli pienillä kierrosnopeuksilla pystyä tuottamaan vähintään kaksinkertainen vääntömomentti kuin testattava laitteisto tuottaa. Näin testattavan laitteiston liike saadaan luotettavasti pysäytettyä. Valittavan taajuusmuuttajan tulee pystyä tuottamaan vähintään vastaava teho, jonka käytettävä moottori pystyy muuntamaan mekaaniseksi vääntömomentiksi.

Moottoriksi valittiin 690 rpm:n, 0,55 kW:n oikosulkumoottori, jonka napaluku on 8. Moottori on VEM-motorsin valmistama ja tyypiltään K21R 90 L8. Moottori kykenee tuottamaan noin 2Nm:n vääntömomentin, joka vastaa noin 10Nm:n vääntömomenttia moottoriakselin ulkoreunalla, johon kaikki kuormitusmomentit on suhteutettu. Tämä riittää pysäyttämään testattavan ovioperaattorin ohjaaman moottorin liikkeen luotettavasti.

Kun moottori saatiin mitoitettua ja valittua, voitiin kartoittaa tarvittava taajuusmuuttaja, jonka ensimmäiseksi perusteeksi tuli teho, joka vastaa moottoria. Tämän lisäksi, koska toimintaa tapahtuu pääosin lähellä 0-nopeutta ja muutoinkin pienillä kierrosnopeuksilla, tuli kyseeseen suljetun silmukan (Closed Loop) ohjaus, jotta momenttisäätö voitiin toteuttaa

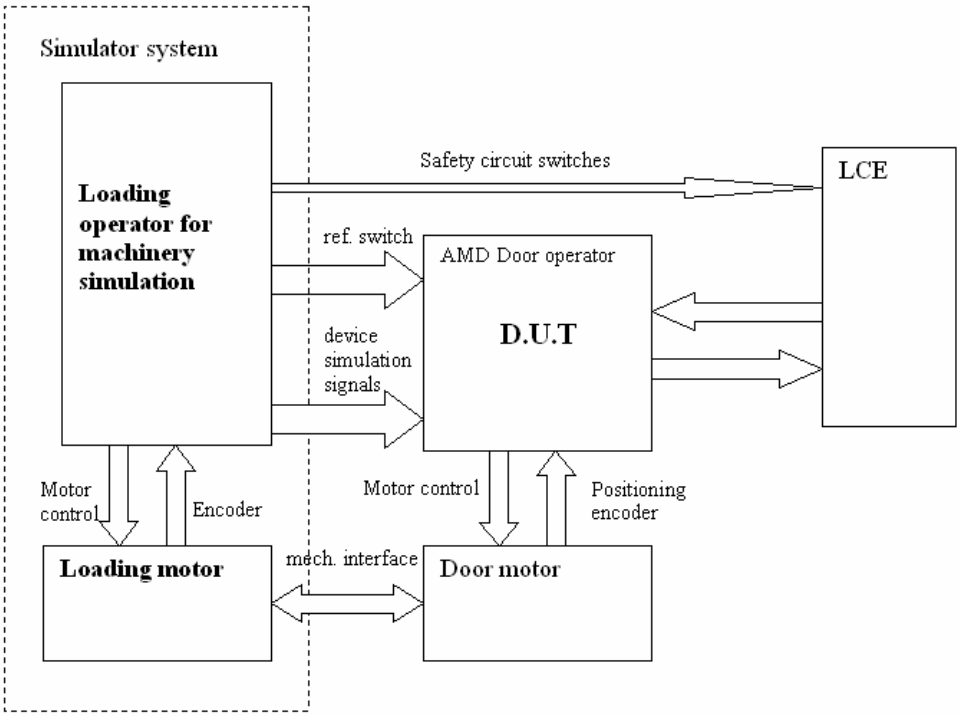
luotettavasti. Edellisten lisäksi säätö piti voida suorittaa analogisella signaalilla ja taajuusmuuttajaa piti pystyä hallitsemaan digitaalisen I/O-liitynnän kautta seuraavin signaalein:

- Käynnistyskomento (Input)
- Ohjainkortin toimintahäiriö, jolloin moottori ei saa käynnistyä (Input)
- Ohjainkortille tieto taajuusmuuttajan vikatilasta (Output)
- Ohjainkortin automaattinen taajuusmuuttajan vian kuittaus (Input)
- Analoginen ohjaus-signaali momenttiohjeelle.
- Closed Loop –takaisinkytkentä pulssienkooderilla

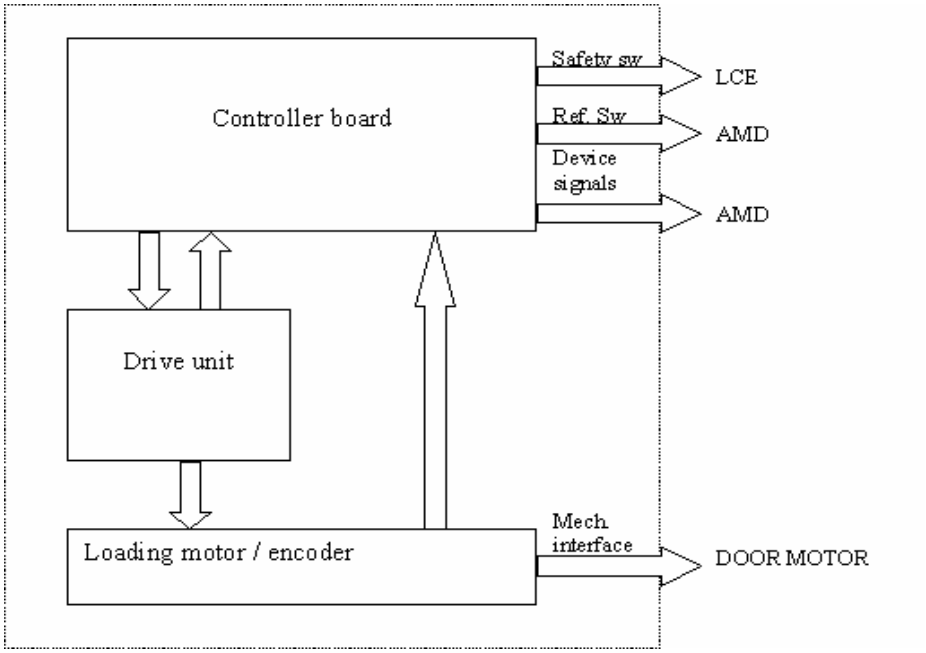
Lisäksi alkuperäisen suunnittelun mukaan piti olla myös käyntikomennot eteen ja taakse suunnille, jotka kuitenkin jäivät tarpeettomiksi työn edetessä.

Vähintään 550 W:n tehontuotto, Closed Loop–ohjaus ja tarvittavat I/O-liitynnät tuottivat valintamahdollisuuksiksi ABB:n valmistaman ACS 350–sarjan tai Vacon NXP–sarjan taajuusmuuttajat. Konsultoituani molempia valmistajia ja käyttäen hyväksi omaa tietämystäni valitsin käytettäväksi taajuusmuuttajaksi Vacon NXP-sarjan pienimmän mallin Vacon NXP00035A2, jonka teho-alue ja muut ominaisuudet vastasivat tarpeita. Lisäksi Vaconin mallissa käytetään tarkkaa vektorisäätöä koko nopeusalueella /2/.

Kokonaisen simulointilaitteiston ja siihen liittyvien ohjauksien lohkokaaviokuva on esitetty kuvassa 3.1.1., tässä kuitenkin keskitytään laitteistoa ohjaavan ohjauskortin suunnitteluun ja sen toteutukseen, jonka signalointiin ja ohjaukseen liittyvä lohkokaavioesitys on esitetty kuvassa 3.1.2..



Kuva 3.1.1. Testauslaitteiston lohkokaavio



Kuva 3.1.2. Simulaattorin sisäinen signalointi

4 OHJAINKORTIN SUUNNITTELU

4.1 Yleistä

Toiminnallisen määrittelyn aikana päädyttiin ratkaisuun, jossa laitteen hallinta tapahtuu mikroprosessorin ohjauksella. Näin laitteesta saadaan mahdollisimman monipuolinen ja sen testausympäristön hallinta on helppoa myös loppukäyttäjälle. Myös laitteiston tiedonkeruuominaisuudet vaativat prosessoriohjausta.

Ensimmäisellä käyttökerralla laitteisto kalibroidaan manuaalisesti säätämällä taajuusmuuttajan momenttiohjeen nollakohta ja asettamalla maksimivoimat molempiin suuntiin, joilla kuormittavan moottorin momentti pysäyttää testattavan laitteen moottorin. Kalibrointi tehdään komponenttitoleranssien vaikutusten kompensoimiseksi.

4.2 Mikroprosessori

Mikroprosessorin valintakriteerit laitteen ohjauskortille olivat suora laskentateho, jotta simulointilaskenta ja mittaustaajuus saadaan riittävän nopeiksi. Lisäksi prosessorin valintaan vaikutti fyysinen rakenne, minkä vuoksi turhia komponentteja haluttiin välttää. Edellisten lisäksi myös prosessorin I/O –liityntöjen määrä vaikutti valintaan huomattavasti.

Koska ylimääräisiä oheiskomponentteja haluttiin välttää, tuli kyseeseen ohjain jossa on sisäänrakennettuna mahdollisimman suuri määrä tarvittavia muisteja (Flash, EEPROM, RAM), näin ollen valittiin mikro-ohjain tyyppinen prosessori.

Atmelin AVR-piiriperhe on CMOS-tekniikalla valmistettu 8-bittinen RISC-tyyppinen mikro-ohjain, joka rakenteensa puolesta suorittaa yhden konekäskyn yhdellä kellojaksolla. Tämän lisäksi AVR-sarjan mikro-ohjaimet sisältävät oman Flash-muistin, sisäisen EEPROM-muistin, RAM-muistin ja A/D-muuntimen sekä muita ominaisuuksia.

Lisäksi oma tietämys AVR-sarjan piireistä antoi lisäpontta AVR-sarjan ohjaimen valintaan.

Koska laitteen tulee suoriutua niin muistikortin käytöstä kuin muusta I/O-liityntöjä vaativasta työstä, valittiin ohjaimeksi AVR-perheen uudentyyppinen Atmega64. Kyseinen mikro-ohjain sisältää riittävän määrän I/O-liityntöjä (53), Flash-muistia 64kB, jonka arvioin riittävän, sekä SRAM 4kB käyttömuistia, joten RAMissa voidaan käsitellä jo huomattavan kokoisia tietorakenteita. Piirin maksimikellotaajuus on 16MHz, eli laskentateho on noin 16 MIPS:ä /3/.

Luotettavaan mittaustiedon keruuseen tarvitaan noin 100 mikrosekunnin aikaikkuna. Koska mittaukset pitää suorittaa eriaikaisesti, ikkuna ei voi olla kovin suuri, jotta mittaustuloksien voisi olettaa olevan samanaikaisia.

Siten edellisestä laskien prosessorin laskentatehoksi tarvitaan 2 MIPS:ä (Million Instructions per second)

Pahimmassa A/D-muunnos –tapauksessa tarvitaan yhden mittauksen suoritukseen 18 kellojaksoa ja koska mittauskohteita on 7, analoginen takaisinkytkentä taajuusmuuttajalta sekä kolmen moottorivaiheen virta ja jännitetiedot, saadaan tarvittavaksi laskentatehoksi prosessorille $18\,kellojaksoa \cdot 7 = 126\,kellojaksoa$, josta voidaan laskea tarvittava laskentateho:

$$\frac{1}{100\,\mu s} \cdot 126\,kellojaksoa = 1\,260\,000\,IPS \Rightarrow 2\,MIPS$$

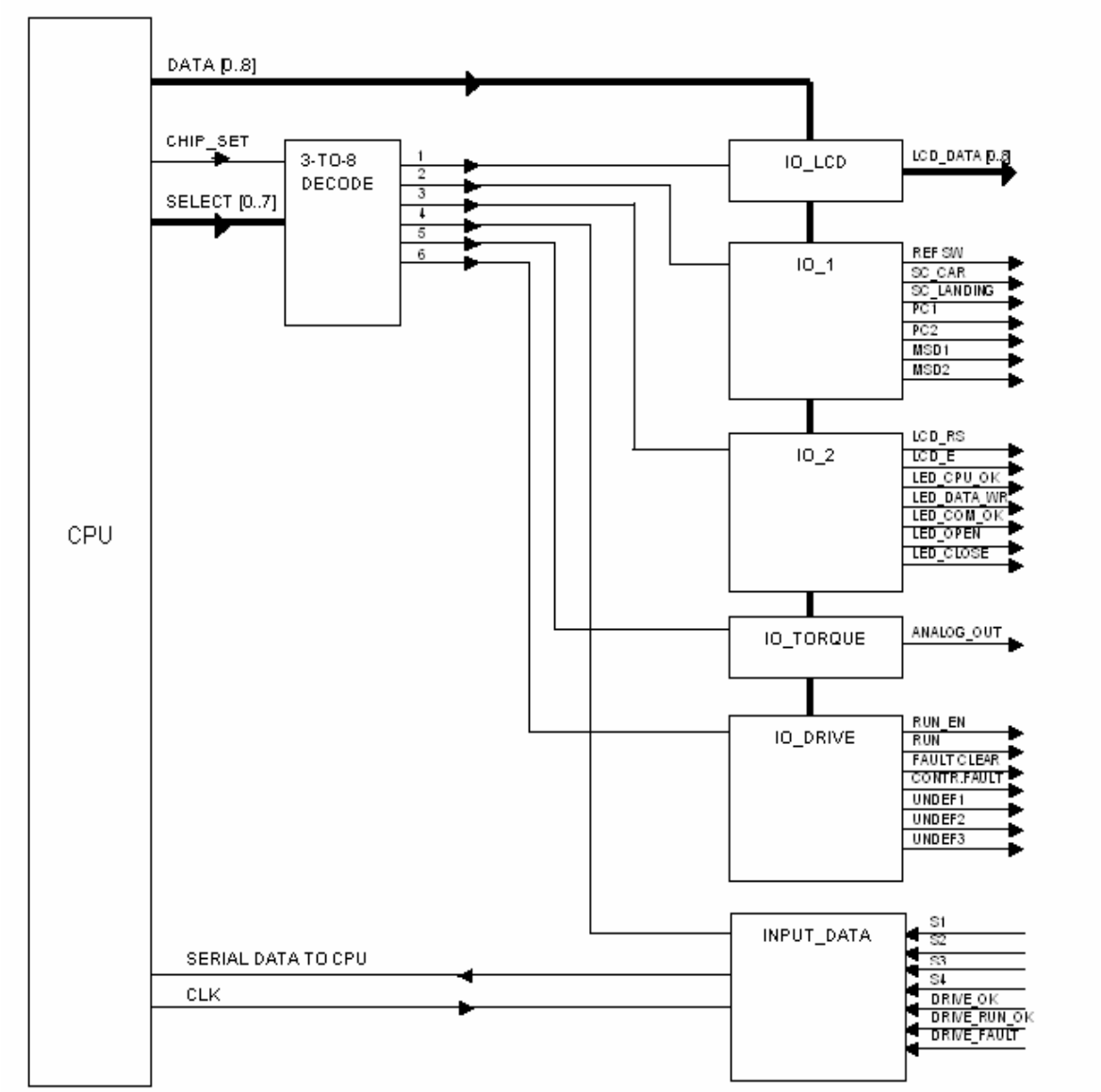
Näin ollen riittäisi 2 MHz:n kellotaajuus. Tämä kuitenkin tarkoittaisi että prosessoritehosta puolet käytetään mittaustiedon keruuseen. Suunnittelun alkuvaiheessa ei tiedetty tarkalleen, miten paljon laskentatehovaatimusta saadaan simuloinnista, jossa käsitellään hyvinkin suuria lukuja ja myös liukulukuja. Tästä kuitenkin voidaan päätellä, että simulointilaskentaan tarvitaan laskentatehoa huomattavasti enemmän kuin mittaustiedon keruuseen. Laskenta tulee saadaan suoritettua siten, ettei se häiritse muuta toimintaa. Näin ollen mikro-ohjaimen kellotaajuudeksi valittiin 16MHz, jotta tarvittava aikamarginaali mittausnopeuteen yhdessä muiden toimintojen kanssa saavutetaan.

4.3 Laajennettu I/O-väylä

Suunnittelun edetessä I/O-liityntöjä täytyi kuitenkin laajentaa, jolloin mikroprosessorin 1 I/O-portti otettiin dataväyläksi ja toisen portin I/O-liitynnät ohjaamaan dataväylän laajennuspiirejä. Dataväylä toimii yhdensuuntaisena väylänä ulospäin ja laajennettu sisääntulo tehtiin sarjamuotoisena siten, että laajennuspiirinä käytettiin rinnakkaisottoista siirtorekisteriä 74HC165. Laajennuspiireiksi output-väylään valittiin 74HC573-piiri.

Väyläpiirejä hallitaan erillisellä 3-to-8-multiplekseripiirillä 74HC138. Näin ollen yhteensä 14:lla mikroprosessorin I/O-liitynnällä, joista 8 toimii databitteinä, saatiin tehtyä I/O-laajennuksia siten että saatiin 8 bittiä input-biteiksi ja 40 bittiä output-biteiksi. Näin toteutettuna liityntää on mahdollisuus laajentaa edelleen siten, että input-tulopiirejä voidaan lisätä rajattomasti, mutta output-signaaleja voidaan kytkentään lisätä enintään 8, jotta väylän hallintapiireihin ei tarvitse koskea.

Laajennetun I/O-väylän signaalikaavio on esitetty kuvassa 4.3.1.



Kuva 4.3.1. Laajennettu I/O-väylä

4.4 Käyttöliittymä ja hallinta

Paikallisen käyttöliittymän muodostavat neljä hallintanäppäintä, LCD-näyttö ja indikointi-LEDit eri toimintatilojen määrittämiseksi. Laitteen hallinta voidaan suorittaa joko paikallisesta käyttöliittymästä tai etähallinnan kautta suoraan tietokoneen näytöltä RS-232-liitynnän kautta.

4.4.1 LED-indikointi

Toimintatilojen indikointi tapahtuu LED-merkkivaloilla, joita ohjataan lähinnä laajennetun Output-väylän kautta U9-piirin /1/ kautta (5kpl). Tämän lisäksi ohjainkortilla on indikointi taajuusmuuttajan tilan toteamiseen ja mikro-ohjaimen ohjelmointiin. Käyttöliittymäledien kuvaukset ovat taulukossa 4.4.1.1. ja niiden toiminnallinen kuvas taulukon alla.

Nimi	Osanumero	Väri	Virta / mA
LED_DATA_WR	D23	Punainen	5
LED_ISP	D24	Punainen	5
LED_-15V	D27	Vihreä	5
LED_5V	D28	Vihreä	5
LED_15V	D31	Vihreä	5
LED_CPU_OK	D32	Vihreä	5
LED_COM_OK	D33	Vihreä	5
LED_DRIVE_OK	D34	Vihreä	5
LED_CLOSE	D47	Vihreä	5
LED_OPEN	D48	Vihreä	5
LED_24V	D51	Vihreä	5

Taulukko 4.4.4.1. Käyttöliittymä-ledit

Taajuusmuuttajan tilaindikointi, D34 (LED_DRIVE_OK) /1/ on samassa signaalissa josta tieto mikro-ohjaimelle luetaan. Tietosisältö tässä signaalissa kertoo taajuusmuuttajan olevan valmis toimimaan tai sen ettei se ole toimintavalmis.

Mikro-ohjaimen ohjelmoinnin tila-indikointia hallitaan mikro-ohjaimen ISP-ohjelmointiliittimen kautta, eli ohjauskortti ei pysty vaikuttamaan sisäisesti tämän LEDin toimintaan. Tämä indikointi on tarkoitettu asennettaksi vain laitteen prototyyppivaiheessa.

Laajennetun output-väylän kautta hallittavat indikoinnit kertovat Oven suuntatiedon (LED_OPEN tai LED_CLOSE) LEDien D47 ja D48 /1/ avulla.

D32 (LED_CPU_OK) kertoo mikroprosessorin toiminnan olevan kunnossa kun se vilkkuu noin 1Hz:n taajuudella.

D23 (LED_DATA_WR) palaa silloin, kun tietoa kirjoitetaan muistikortille. Tietoa kirjoitetaan muistikortille silloin kun siihen on prosessoriaikaa tai muistipuskuri on täynnä. Tällöin kaikki muut paitsi taajuusmuuttajan ohjaus- ja simulointitoiminteet ovat poissa käytöstä, jotta tieto saadaan kirjoitettua muistiin mahdollisimman nopeasti. Jos muistipuskuri täyttyy, eikä prosessoriaikaa ole, ohjelman muut toiminteet keskeytyvät siksi aikaa, kunnes puskurista on kirjoitettu muistiin tietoa ja puskurissa on tilaa.

D33 (LED_COM_OK) palaa silloin, kun sarjaliikennelinkki on päällä ja yhteys toimii.

Näiden tilatietoledien lisäksi kortilla on käyttöjänniteindikoinnit +24VDC, +15VDC, +5VDC ja -15VDC käyttöjännitteille.

Ohjainkortin ledien lisäksi laite sisältää Vacon taajuusmuuttajan oman diagnosoinnin ja ohjainkortin lisädiagnostiikan, kuten tilasanat ja pulssienkooderin toiminta voidaan todeta ulkoisen käyttöliittymän kautta.

4.4.2 Hallintanäppäimet

Hallintanäppäimet on kytketty mikro-ohjaimelle laajennetun input-väylän kautta, joka luetaan sarjamuotoisesti.

Laajennettu input-väylä on toteutettu ulkoisella siirtorekisterillä U6 /1/ (Paraller-to-Serial), jolloin on säästetty mikro-ohjaimen I/O-liitäntöjä. Tähän väylään on kuitenkin kytketty vain ne input-linjat jotka eivät ole kriittisiä toiminnan kannalta ja niitä voidaan lukea harvemmin. Siirtorekisteri toimii MSB-first-periaatteella, eli eniten merkitsevä bitti tulee rekisteristä ulos ensimmäisenä. Näin ollen mikro-ohjaimen tilatietomuuttujassa hallintanäppäimet sijaitsevat biteissä 0-3 (S1..S4).

Hallintanäppäinten tila luetaan 10ms:n välein ja ne on asetettu toimimaan laskevan reunan aikana, jotta mahdollinen näppäinvärähtely saadaan eliminoitua ilman erilliskomponentteja ohjelmallisesti. Hallinta-näppäimet on kytketty laajennetun I/O-väylän INPUT_DATA-piiriin. Laajennetun I/O-väylän signaalikuva on esitetty kuvassa 4.3.1.

Samanaikaisesti saadaan tieto myös taajuusmuuttajan tilasta, sen käynnissä olosta ja siitä, onko taajuusmuuttaja vikaantunut. Näiden tietojen käyttö kuvataan myöhemmin luvussa 4.5 Taajuusmuuttajan ohjaus.

4.4.3 LCD-näyttö

LCD-näyttö toimii käyttöliittymän päänäyttönä, josta voidaan lukea valintojen ja tilan mukaan valikkotiedot, parametrointitiedot ja valvontatiedot.

Valikkonäytössä LCD-näytöllä näytetään senhetkinen sijainti valikossa, parametrinäyttö kertoo asetetun parametriarvon ja valvontatietona voidaan näyttää oven sijainti, hetkellinen nopeus tai taajuusmuuttajan takaisinkytkentätieto moottorin momentista.

LCD-näyttöä ohjataan 8-bitin ohjausmuodolla laajennetun output-liitynnän kautta jossa databitit kulkevat piirin U8 kautta ja ohjaussignaalit Enable (LCD_E) ja Register select (LCD_RS) suoraan mikro-ohjaimen I/O-liitynnöistä. LCD-näytön päivitysväli on niin hidas, ettei sen lukutoimintoja välttämättä tarvita varmistamaan LCD-näytön valmiutta ottaa dataa vastaan. LCD-näytön hallinta on esitetty laajennetun I/O-väylän signaalikuvassa

4.4.4 Etähallinta

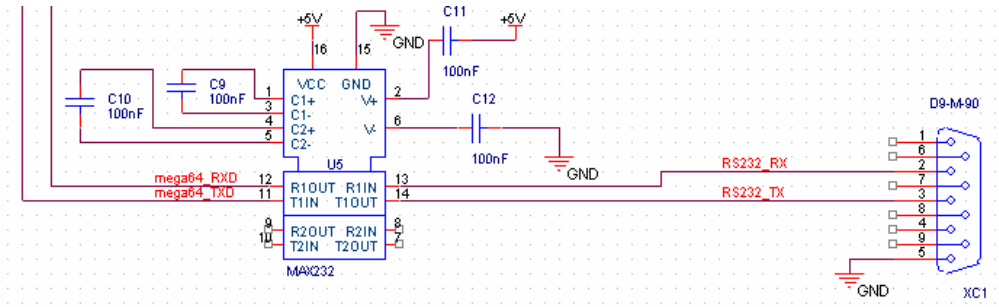
Etähallintaliityntänä toimii RS-232-liityntä 9600bps nopeudella ja käytössä on 8 tietobittiä ja 1 lopetusbitti. Laitteessa ei käytetä vuon hallintaa eikä muita kättelytoimintoja.

Tämä liityntä on EIA:n RS232 standardin mukainen /2/. RS-232 liitynnän kytkentäkuva on esitetty kuvana 4.4.4.1.

Etähallintaohjelmana toimii normaali standardin mukainen pääteohjelma, johon tulostetaan tarvittava hallintatieto suoraan ohjainkortilta. Etähallinta sisältää täysin samat ominaisuudet kuin paikallinen hallinta ja lisäksi tiedonkeruutilan, jossa kaikki mittaustiedot tulostetaan mitattaessa suoraan RS-232-liityntään ja tieto voidaan kerätä pääteohjelmalla myöhempää käyttöä varten.

Tiedonkeruuhallinnan lisäksi etähallintaohjelmisto mahdollistaa myös parametrointitietojen muuttamisen suoraan ohjainkortilla olevaan EEPROM-muistiin ja muistikortin lukemisen tietokoneelle suoraan liitynnän kautta.

RS-232-väylä on toteutettu valmiilla muunninpiirillä MAX232 (U5) ja tämän tarvitsemilla oheiskomponenteille (C9-C12) ja mikro-ohjaimen sisäisillä RS-232-rekistereillä /1/ /3/. Laitteistoon liitytään terminaaliohjelmalla ja käyttämällä standardin mukaista ristiinkytkettyä RS-232-kaapelia.



Kuva 4.4.4.1. RS-232 liityntä

4.5 Taajuusmuuttajan ohjaus

Vacon NXP-taajuusmuuttajassa on vakiona 6 ohjelmoitavaa digitaalista tuloa joiden käyttöjännite on 18..30VDC ja yksi digitaalinen open collector-lähtö sekä lisäksi kaksi ohjelmoitavaa relelähtöä.

Vakiona taajuusmuuttajassa on myös yksi analoginen ohjausliityntä ja yksi analoginen ohjelmoitava lähtöliityntä /2/.

Tässä luvussa kuvataan taajuusmuuttajan ohjaaminen ohjainkortin näkökulmasta.

Liityntää ohjataan laajennetun I/O-väylän kautta siten kuin, se on esitetty kuvassa 4.3.1.

4.5.1 Digitaaliset liitynnät

Vacon-taajuusmuuttaja tarvitsee logiikan toimintaan 24VDC käyttöjännitteen, joka tuotetaan samasta lähteestä kuin ohjauskortin käyttöjännitteet, mutta ohjauskortin loogisen toiminnan käyttöjännite on 5VDC, joten positiivisten käyttöjännitteiden galvaaninen erotus on tehty optoerottimia käyttäen.

Output-signaaleissa on käytetty LCA110LS optoerotinta ja 100 ohmin suojavastusta, jotta kytkentätilanteessa mahdolliset transientti-piikit hidastuvat. Nämä eivät ole täysin välttämättömiä, sillä LCA110LS optoerotin sisältää virranrajoituspiirin ja omaa suuren impedanssin /5/. LCA110LS optoerotin on sopiva mikroprosessoriohjaukseen ja sen hyvä virranantokyky takaa luotettavan toiminnan testilaitteessa. Virranantokyky on tarkoituksellisesti mitoitettu luotettavuuden vuoksi huomattavasti korkeammaksi kuin olisi tarve.

Samaa käytäntöä on noudatettu, kun taajuusmuuttajan digitaalinen lähtösignaali on muutettu sopivaksi ohjauskortin input-logiikkaan. Tällöin käytössä on TLP181 optoerotin, joka toimii moitteetta, kun tarvittava lähtövirta on pieni. Tätä signaalia käytetään taajuusmuuttajan toiminnan tarkistamiseen (DRIVE_OK) /1/.

Taajuusmuuttajan relelähdöt on kytketty siten, että niihin on ohjauskortilta suoraan tuotu 5V:n looginen käyttöjännite ylösvetovastusten kautta, joten näitä voidaan käyttää suoraan laajennetun Input-väylän liitäntöihin samaan tapaan kuin hallintanäppäimiä.

4.5.2 Analogiset liitynnät

Taajuusmuuttajan analoginen sisäänmeno on asetettu ottamaan vastaan 4..20mA –virtaviestiä ja sitä käytetään momenttiohjeena moottorin ohjaukselle. Momentin skaalaus taajuusmuuttajalla on -100%...100%, mikä vastaa virtaviestin arvoa, eli tällöin 0% ohjearvo on 12mA.

Momenttiohje muodostetaan laajennettuun Output-väylään liitetyllä D/A –muuntimella AD7524 (U10), joka on 8-bittinen muunnin. AD-muuntimella muodostetaan 0...5V jännite joka muunnetaan U3D operaatiovahvistimella /1/, transistorin Q9 ja muutaman oheiskomponentin avulla 4..20mA virtaviestiksi.

4...20mA virtaviesti on käytössä turvallisuussyistä. Jos johdin katkeaa, vaikka se ei ole todennäköistä, taajuusmuuttaja ei tällöin käynnisty. Jos käytössä olisi 0..20mA virtaviesti, johdon katkeaminen aiheuttaisi momenttiohjeen hallitsemattoman nousemisen -100%:iin jolloin vapaasti pyörivä moottori tuhoaa jännitteen noustessa testattavan ovioperaattori-kortin.

Taajuusmuuttajan analoginen lähtö on ohjelmoitu käyttämään myös 4...20mA virtaviestiä ja tätä syötetään ohjauskortilla olevaan kuormavastukseen (R52 -100ohm) /1/. Vastuksen yli muodostuvasta jännitteestä 0,4...2V muunnetaan differentiaalivahvistimen avulla jännitearvoksi 1..5V, joka tästä muunnetaan prosessorin sisäisen A/D-muuntimen avulla /3/ jatkokäsittelyä ja tiedon tarkistusta varten.

4.6 Paikannus ja massan hitausmomentin simulointi

Oven paikannukseen ja nopeuden mittaamiseen käytetään pulssienkooderia. Samaa enkooderia käytetään koko laitteiston mittakaavassa myös taajuusmuuttajan takaisinkytkennässä, joten enkooderin signaalit on viety ohjauskortin kautta myös taajuusmuuttajalle.

Signaalien muuntamiseen käytetään ohjauskortilla HCPL-0611 optoerotinta, jonka nopeus ja luotettavuus riittävät kyseiseen sovellukseen /6/ /7/.

Pulssienkooderi tuottaa 90 asteen vaihesiirrossa olevat pulssit kanaville 1 ja 2. Signaaleja käytetään ohjauskortilla siten että kanavasta 1 luetaan nopeustieto, ja kanava 2 kertoo suunnan. Jos kanavan 1 laskevalla reunalla kanavan 2 signaali on loogisessa 0-tilassa, on ovi kulkemassa auki-suuntaan. Jos taas kanavan 1 laskevalla reunalla kanavan 2 signaali on loogisessa 1-tilassa, tulkitaan että ovi on liikkumassa kiinni-suuntaan.

Taajuusmuuttaja on parametroitu lukemaan enkooderin signaalia täysin vastaavasti kuin ohjauskortti.

Ohjauskortilla kanavan 1 signaalia käytetään tuottamaan keskeytys jokaisella laskevalla reunalla, jolloin lisätään tai vähennetään pulssilaskurin arvoa oven suunnan mukaan.

Tämä arvo luetaan 20ms:n välein, kuten jäljempänä ohjelmiston kuvauksessa on esitetty. Tällöin myös tallennetaan entinen arvo. Ohjelmiston kuvauksessa, kappaleessa 5.9, on kerrottu, miten hitausmomentin suuruus lasketaan ohjelmassa ja miten se suhteutetaan oven hetkelliseen kiihtyvyyteen. Myös hitausmomentin simulointilaskenta kuvataan jäljempänä, kappaleessa 5.9. Tämän jälkeen se muunnetaan 8-bittiselle D/A-muuntimelle sopivaksi, jotta tiedosta voidaan tuottaa oikean arvoinen milliampeeriviesti.

Näin ollen momenttiohje on prosentuaalisesti suhteutettu minimi- ja maksimiarvon välille, kuten se on käsitelty myös taajuusmuuttajassa. Kun momentti on suhteutettu mekaaniseen rakenteeseen sen maksimiarvo on 9,5 Nm, kun testattavan ovioperaattorin ja moottorin tuottama momentti tällöin on 2,6 Nm. Suhteutettaessa vääntömomenttia prosentuaaliseksi on maksimiarvona käytetty lukua 10 Nm. Näin ollen momenttiaskel muodostuu siten että luku 127 tarkoittaa 0%:a ja luku 255 100%:a. Tällöin siis momenttiaskel on 127 / 100 eli 1,27% mikä tarkoittaa 10Nm:n vertailuarvoon nähden 0,12 Nm:n askelta. Tästä muodostetaan virtaviesti taajuusmuuttajalle, kuten aiemmassa kappaleessa 4.5. on esitetty.

4.7 Ovioperaattorin I/O-hallinta

Ovioperaattori ja hissin ohjausjärjestelmä tarvitsevat muutamia I/O-signaaleja oven tilan tunnistamiseen, joten myös nämä signaalit täytyy simuloida oikealla tavalla.

Oven I/O-signaalit toimivat myös laajennetun output-väylän kautta jossa niitä ohjataan piirin U7 avulla /1/.

Tärkein näistä on tunnistekytkin (REF_SW), jonka ovioperaattori tarvitsee, jotta voidaan tunnistaa oven mekaaninen asento, kun on se sulkeutunut täysin, mutta oven mukaanottajan jousi on vielä lepoasennossa. Tätä varten tarvitaan ohjaus sähköiselle magnetoinnille, joka on muunnettu 24V:n signaaliksi FETin Q3 avulla, ja liittimen yli on asetettu myös suojadiodi D11 virran katkeamisesta aiheutuvien negatiivisten jännitetransienttien johtamiseksi käyttöjänniteeseen ja näin ollen niiden suodattamiseen. Ilman tätä diodia Q3:n rikkoutumistodennäköisyys kasvaisi huomattavasti.

Signaalit (SC_CAR ja SC_LANDING) /1/ ovat rele-lähtöjä, sillä niiden mitoitus täytyy tehdä 230VAC jännitteelle ja 2A:n virralle. Näitä releitä ohjataan transistorikytkinten (Q2 ja Q8) kautta. SC_LANDING-signaali on aktiivinen yhtä aikaa REF_SW-signaalin kanssa ja SC_CAR silloin kun ovi on täysin kiinni ja mukaanottaja täysin auki.

Testattava ovioperaattori sisältää myös oven mekaanisten turvalaitteiden hallinnan joten täysipainoisen mekaanisen simuloinnin tuottamiseksi myös nämä signaalit pitää pystyä simuloimaan. Signaalit PC1 ja PC2 /1/ on tehty open collector-tyyppisesti siten, että testattavassa ovioperaattorissa voidaan käyttää positiivista tai negatiivista logiikkaa sen asetuksista riippuen. Signaalit MSD1 ja MSD2 puolestaan toimivat vain negatiivisella logiikalla, sillä operaattoreissa toiminta on vain yksipuoleista.

4.8 Tiedonkeruu ja tallennus

Laitteiston toiminnallisen määrittelyn mukaan testilaite kerää ja tallentaa mittaus- ja muut tarpeelliset tiedot muistikortille testauksen aikana.

Mittaustiedot tallennetaan normaalille CompactFlash-muistikortille, johon tieto voidaan varastoida suoraan sellaiseen tekstimuotoon, että se voidaan lukea sieltä PC:llä. CF-kortin liitäntä mikro-ohjaimen I/O-väyliin tapahtuu suoraan liittimen XCF1-kautta. Tiedon tallennusta käsitellään tarkemmin luvussa 5.Ohjauskortin ohjelmisto.

Osa tallennettavasta tiedosta on ohjainkortin itse laskemaa tietoa liittyen oven hetkelliseen nopeuteen, hetkelliseen kiihtyvyyteen ja momenttiohjeeseen. Kuitenkin ohjainkortti mittaa myös testattavan operaattorin moottorille syöttämää jännitettä ja virtaa, jotka tallennetaan analysointia varten.

Ongelmalliseksi mittauksen tekee sisäinen A/D-muunnin, joka toimii vuorotteluperiaatteella, eli tietoja ei saada täysin samanaikaisesti muistiin, mutta niitä voidaan kuitenkin käyttää analysointiin edellä esitetyn perusteella, missä todettiin että prosessorin maksimiaika A/D-muunnoksen suoritukseen on 180 kellojaksoa, jonka kesto 16MHz:n kellotaajuudella on 11 mikrosekuntia. Näin ollen kaikkien arvojen lukeminen

kestää yhteensä 7 x 11 mikrosekuntia = 77 mikrosekuntia, jolloin aikaikkuna jää hyvin pieneksi. Tällöin kuitenkin nopeat transienttipiikit jäävät huomaamatta. Mittaustiedot ovat kuitenkin luotettavia, jos halutaan tarkistaa moottorijännitteitä ja virtoja yleisesti testauksen aikana.

Moottorivirrat ja jännitteet mitataan ohjauskortilla, jonka kautta ovioperaattorin ja ovimoottorin väliset moottorin vaihejohtimet johdetaan.

Moottorivirrat mitataan tähän käyttöön suunnitellulla VAC N4644-X400-virta-anturilla jokaisesta moottorivaiheesta erikseen. Virta-anturi on asetettu toimimaan 1:2000:een muuntosuhteella jolloin esimerkiksi 30 ampeerin virralla, joka on maksimi-virta joka mitataan, ulostuloon saadaan 30A/2000 = 15mA:n virta /10/. Joka tällöin vastaa 250 ohmin kuormaan syötettynä 3,75V:n jännitettä. Näin ollen mitattava alue on -15...15mA. Tämä virta-alue muunnetaan operaatiovahvistinkytkennän avulla 0..5V:n jänniteviestiksi, joka voidaan lukea mikro-ohjaimen sisäisellä A/D-muuntimella tiedon tallennusta varten. Tällöin 0V vastaa -30A:n virtaa johtimessa ja 5V vastaavasti 30A:n moottorivirtaa.

Vastaavasti moottorivaiheiden jännitteet mitataan mikro-ohjaimen A/D-muuntimella tiedon tallennusta varten. Tämä tapahtuu suoraan jännitejakokytkennän ja jänniteseuraajakytkennän avulla moottorin vaihejohtimesta. Mittausresistanssi on niin suuri, ettei se haittaa moottorin varsinaista toimintaa, eikä toisaalta kuormita operaattoria niin, että siitä olisi näkyvää haittaa toiminnalle. Mittauksessa jännite vaimennetaan ensin 1/10-osaan, minkä jälkeen se luetaan jänniteseuraajakytkennällä ja signaali vaimennetaan vielä 1/5-osaan alkuperäisestä jolloin saadaan suoraan verrannollinen jännitesignaali luettavaksi mikro-ohjaimelle ja tallennettavaksi jatkokäsittelyä varten.

4.9 Käyttöjännitteet

Ohjauskortti tarvitsee toimiakseen ulkopuoliset käyttöjännitteet +24VDC ja -15VDC, jotka muodostetaan kaupallisilla Weidmüllerin valmistamilla hakkuriteholähteillä.

Kortin sisäiset käyttöjännitteet muodostetaan näistä lineaarisilla regulaattoripiireillä ja operaatiovahvistimella. Testilaitteessa on pyritty mahdollisimman tasaisiin käyttöjännitteisiin, osin tehonkulutuksen kustannuksella. Tehon kulutus testaus-laiteympäristössä ei ole niin kriittistä kuin esimerkiksi kannettavissa laitteissa.

Ohjauskortin teholähteen jännitenapaisuuden väärinkytkentä on estetty diodien D49 ja D50 avulla, ja käyttöjännitesyöttöön on kytketty erilliset sulakkeet.

+24VDC:n käyttöjännitteestä tehdään kortin positiiviset käyttöjännitteet lineaarisilla regulaattori-piireillä. +15VDC:n käyttöjännite muodostetaan 7815-regulaattorilla 24VDC:n jännitesyötöstä ja siitä edelleen muunnetaan lineaarisella regulaattoripiirillä 7805 +5VDC:n käyttöjännitteet logiikkapiireille sekä mikro-ohjaimelle.

Tämän lisäksi +5VDC:n käyttöjännitteestä tehdään 2,5VDC:n referenssijännitetaso jännitteenjakokytkennällä ja operaatiovahvistimella, jonka käyttöjännitteenä käytetään +15VDC:n jännitettä, jotta tehonsyöttö jakautuisi paremmin koko teholähteelle. 2,5VDC:n referenssijännitettä ei juurikaan kuormiteta, joten sen lähteenä voidaan käyttää operaatiovahvistimen lähtöliitäntää.

5 OHJAINKORTIN OHJELMISTOSUUNNITTELU

Ohjelmiston voidaan sanoa olevan simulointilaitteiston sydän, koska ohjainkortti on tehty mikroprosessoripohjaisesti. Kaikki loogista toimintaa ja päättelyä vaativa toiminta on sisällytetty näin ollen ohjelmistoon. Laitteen määrittelyvaiheessa päätettiin, että laitteen tulee kyetä simuloimaan erilaisia ovia kulloisenkin profiilin mukaisesti, minkä lisäksi ohjelmistoon on sisällytetty vaadittu mahdollisuus oman profiilin luomiseen.

Vastavoiman muodostus eli oven mekaanisen rakenteen aiheuttaman hitausmomentin simulointi perustuu mitattuun akselin pyörimisnopeuteen, joka muunnetaan virtuaaliseksi vertikaaliseksi liikkeeksi pulssienkooderin avulla.

Simuloinnin lisäksi ohjelmistoa tulee kyetä hallitsemaan myös ulkopuoliselta tietokoneelta sarjaliikennelinkin kautta, ja tiedon tallennus muistikortille tapahtuu myös ohjelmiston avulla.

Edellisten lisäksi ohjelmiston avulla hoidetaan myös käyttöliittymä ja paikallinen hallinta.

Testauslaitteiston ohjelmiston tulee täyttää hyvät luotettavuuskriteerit, jotta laitteella voisi tehdä luotettavia, pitkäaikaisia mittauksia.

Ohjelmistosuunnittelu aloitettiin tekemällä ohjelmistosta toiminnallinen määrittely (liite 1).

Ohjelmisto on kirjoitettu ISO C99-standardin mukaisella C-ohjelmointikielellä, siten kuin se on mahdollista kyseistä ohjelmointialustaa ja kääntäjäympäristöä käyttäen. Ohjelmistosuunnittelu ja ohjelmointi on tehty CodeVisionAVR-ohjelmointiympäristössä. Ohjelmoinnissa on käytetty ympäristön tarjoamia valmiita funktioita siten, kuin se on ollut mahdollista. Vakioiden määrittelyssä on pyritty noudattamaan selkeitä merkintätapoja ja lisäksi on käytetty valmiita ohjelmointiympäristön tarjoamia vakioita, jotka kattavat käytettävän mikro-ohjaimen kaikki toiminnot. Mikro-ohjaimen toimintoihin viittaavat vakiot ovat samat ohjelmistossa kuin piirin valmistajan datalehdessä /3/.

5.1 Mikro-ohjaimen alustus

Virran kytkemisen jälkeen mikro-ohjain alustaa itsensä normaaliin tapaan ja hyppää ohjelmalaskurin nolla-arvosta siihen osoitteeseen, johon kääntäjä on muistiin sijoittanut ohjelman alkukohdan.

Itse mikro-ohjaimen alustustoimenpiteinä alustetaan ensin kaikki I/O-linjat, joita mikro-ohjain käyttää. ATmega64 piirillä I/O-liityntöjä on 53 kappaletta, jotka on jaettu kuuteen 8-bittiseen mittaiseen porttiin ja yhteen 5-bittiseen porttiin. Tämän jälkeen alustetaan ajastimet, joista käytössä on 2, yhteensä piirillä on 4 ajastinta, joista kaksi on 16 bitin

mittaista ja loput mitaltaan 8 bittiä. Käytettävä ajastimet ovat Timer0 ja Timer1, joista ensimmäinen on 8-bittinen ja toinen 16-bittinen.

Ajastinten alustamisen jälkeen alustetaan käytettävät ulkoiset keskeytykset sekä niiden että ajastinten keskeytyssignaalien tulkintatilat.

Näiden jälkeen alustetaan A/D-muunnin ja sarjaliikenne-oiminteet.

Alustuksen yhteydessä myös käyttämättömät toiminteet alustetaan siten, että ohjain tulkitsee niiden olevan poissa käytöstä.

Kun varsinainen pohja-alustus on tehty, alustetaan myös kaikki muuttujat niiden asetusten pohjalta, jotka on laitteen edellisellä käyttökerralla tehty ja tallennettu.

5.2 Käytettävät ajastimet ja niiden alustus

Ajastimia käytetään sovelluksen toimenpidehallintaan, jolloin saavutetaan tietty aikaväli esimerkiksi simulointilaskelman tekemiseen ja sen toimenpiteiden tekemiseen. Näin voidaan hallita helposti tapahtumien kulkua ja toisaalta aikapohjainen hallinta tuottaa tehtäville reaaliaikaisuutta, vaikkakaan sovellus ei sisällä erillistä reaaliaikaista käyttöjärjestelmää. Ajastimien tuottamat keskeytykset on priorisoitu siten että suuremmalla prioriteetilla (ensin suoritettava) toimii Timer1, jolla ajastetaan muun muassa nopeusmittaus ja simulointilaskelmat.

5.2.1 8-bittinen ajastin – Timer0

Timer0:aa käytetään luomaan mikro-ohjaimelle keskeytyspyyntö 100 mikrosekunnin välein. 100 mikrosekunnin ajastettua keskeytystä käytetään nopeiden toiminteiden ajastukseen ja luodaan 10ms:n välein tapahtuvien toiminteiden vaatima ohjelmallinen merkkisignaali. Kuvassa 5.2.1. on esitetty 8-bittisen ajastimen alustusrutiini.

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 2000,000 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
ASSR=0x00;
TCCR0=0x02;    //set prescaler for timer. 0x02 = ClockSource / 8
TCNT0=0x38;    //set to OVF every 100us
OCR0=0x00;
```

Kuva 5.2.1. 8-bittisen ajastimen alustusrutiini

8-bittistä ajastinta käytetään muun muassa LCD-näytön vaatimien viiveiden luomiseen ja käyttöliittymänäppäinten lukemisen ajoittamiseen sekä oven rajasimuloinnin korkean voiman ajoituksessa. Ajastimen keskeytysohjelmaan hypättäessä alustetaan Timer0:n arvo aina uudelleen arvoon 0x38. Ajastimen keskeytys tapahtuu aina, kun arvo ylittää arvon 0xFF ja palaa näin arvoon 0x00.

5.2.2 16-bittinen ajastin - Timer1

16-bittisen ajastimen alustus tapahtuu täysin samalla tavalla kuin edellä on kuvattu 8-bittiselle ajastimelle tehtävä alustus. Alustusrutiini on esitetty kuvassa 5.2.2.

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 2000,000 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x02;          //set timer1 prescaler to be ClockSource / 8
TCNT1H=0xF8;          //set OVF to every 1ms
TCNT1L=0x30;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;
```

Kuva 5.2.2. 16-bittisen ajastimen alustusrutiini

16-bittistä ajastinta käytetään tuottamaan ohjelmalle tietosignaali 500ms:n välein sekä ajastamaan oven nopeusmittaus ja simulointilaskelmien tekeminen 20 ms:n välein.

Ajastimen keskeytyksen tapahduttua ajastimen arvo asetetaan takaisin arvoon 0xF830, jotta keskeytys tapahtuu taas 1ms:n päästä.

5.3 Sarjaliikenneväylän alustus

Sarjaliikenneväylä alustetaan käyttämään haluttua bittinopeutta ja lähetettävän ikkunan tietoja mukaan luettuna databittien määrä sekä käytettävien stop-bittien määrä, lisäksi alustetaan tarvittava pariteettitarkistus ja käytettävä sarjaliikennekanava.

Sarjaliikenneväylällä käytettävä nopeus riippuu ensisijaisesti prosessorin kellotaajuudesta. Ohjaimen rakenteen vuoksi myös sarjaliikenteen ohjaus generoidaan perustaajuudesta asetettavalla jakajapiirillä. Jakajassa ei ole mahdollista käyttää desimaalilukuja, siksi tietyillä sarjaliikennenopeuksilla kellosignaalissa esiintyy nopeusvirhettä. Kun käytetään 9600bps bittinopeutta 16MHz:n kellotaajuudella sarjaliikenneväylän nopeusvirhe on 0,2%, mikä on sallittavissa rajoissa. Yli 0,5%:n nopeusvirhettä ei suositella, sillä tällöin tiedonsiirrossa tyypillisesti bittivirheiden määrä kasvaa huomattavan suureksi ja näin ollen tiedonsiirrosta tulee huomattavan epävakaa /3/.

Sovelluksessa käytetään tiedonsiirtoon 9600:n bittinopeutta ja 8 databitin ja yhden stop-bitin tiedonsiirtokehystä. Pariteettitarkistusta ei tehdä. Alustuksen yhteydessä mikro-ohjaimen nopeusrekisteriin UBRR0 kirjoitetaan arvo 0x67 ja ohjaus-rekisteriin UCSR0B arvo 0x98, jolla asetetaan lähetys- ja vastaanottotoiminnot päälle ja vastaanoton

onnistumisen jälkeen annettava keskeytys päälle. Lisäksi alustetaan ohjain toimimaan asynkronisessa toimintatilassa ja käyttämään edellä kuvattua datakehystä kirjoittamalla UCSR0C-ohjausrekisteriin arvo 0x06. Kuvassa 5.3.1 on esitetty sarjaliikenneohjaimen alustusrutiini.

```
// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud rate: 9600
UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

UBRR0H=UBRR >> 8;
UBRR0L=UBRR & 0xFF;
```

Kuva 5.3.1. Sarjaliikenneohjaimen alustus

Viimeiseksi muokataan vielä ohjaimen sarjaliikenteen nopeusrekisteriä laskemalla sen arvo seuraavasti:

$$UBRR = \frac{FCLK}{(16 \cdot BAUD) - 1} \qquad [5.3.1]$$

Jossa FCLK on käytettävä prosessorin kellotaajuus hertseinä ja BAUD haluttu sarjaliikennenopeus. Sovelluksessa käytettävien arvojen kanssa laskien saadaan nopeus-rekisterin arvoksi:

$$UBRR = \frac{16\,000\,000}{(16 \cdot 9600) - 1} = 104{,}16$$

Koska käytetään kokonaislukurekisteriä UBRR, tuloksen desimaali-osa supistuu pois ja näin ollen UBRR0-nopeusrekisterin arvoksi saadaan 0x0068.

5.4 A/D-muuntimen alustus

Mikro-ohjain sisältää 10-bittisen A/D-muuntimen, jolla voidaan lukea yhteensä 8 analogialiityntää multipleksoidun väylän avulla. Näin ollen mittaukset tapahtuvat sovelluksessa peräkkäin, eivät täysin samanaikaisesti. Edellä on kuvattu se nopeus, joka on vaadittu aikaikkunalle, jossa tulokset voidaan vielä katsoa mittauksen kannalta saman-aikaisiksi.

A/D-muuntimen alustus on aloitettu jo edellisessä kohdassa 5.1, jossa alustettiin kyseinen I/O-portti input-tyyppiseksi kirjoittamalla rekisteriin DDRF arvo 0x00 ja edelleen rekisteriin PORTF arvo 0x00.

Tämän jälkeen A/D-muuntimen hallintarekisteriin ADMUX kirjoitetaan arvo 0x00, jolla määritellään A/D-muuntimen referenssijännite. Sovelluksessa käytetään ulkoista referenssiä, joka saadaan piirille jalasta 62 /3/. Tämän jälkeen hallintarekisteri ADCSRA saa arvon 0x87, jolla A/D-muunnin asetetaan toimimaan ja sen kellotaajuudeksi asetetaan 125kHz. A/D-muuntimen alustusrutiini on esitelty kuvassa 5.4.1.

```
// ADC initialization
// ADC Clock frequency: 125,000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: On
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;
SFIOR|=0x10;
```

Kuva 5.4.1. A/D-muuntimen alustusrutiini

5.5 Parametrien alustus

Parametrit alustetaan, kun kaikki mikro-ohjaimen toiminteisiin liittyvä alustukset on tehty.

Parametri-tietorakenne toimii aina käyttömuistissa ja simulointilaskennat, paikannetieto ja muut suureen perustuvat aina parametri-rakenteen sisältämiin tietoihin. Parametri-taulukon arvot luetaan sisäisestä EEPROMista, siten kuin se on luvussa 5.12 kuvattu. Käytössä on x-määrä valmiita oviprofiileja ja käyttäjän oma profiili, jokin näiden profiilien sisältämistä ovikonfiguraatioista luetaan tietorakenteeseen. Luettava profiili määräytyy sen mukaan edellisellä käyttökerralla on käytetty. Tämä tieto on tallennettuna myös sisäiseen EEPROM-muistiin, kohtaan 0x008h.

Parametrit voivat olla tietotyypiltään 8 tai 16 bittisiä. Jos parametriarvo on 8-bittinen, se luetaan muistista suoraan muuttujaan ja se sijaitsee aina EEPROMin parillisessa muistiosoitteessa. 16-bittiset muuttujat luetaan muistista siten että ensin ylempi tavu, joka tämän jälkeen siirretään muuttujassa 8 bittiä vasemmalle ja tämän jälkeen luetaan alempi tavu loogisen tai-operaation avulla muuttujan alempaan tavuun. 16 bittisen muuttujan ylempi tavu on aina EEPROMin parillisessa osoitteessa ja alempi tavu seuraavassa parittomassa osoitteessa. Lukutapahtuma on kuvattu kuvassa 5.5.1.

```
//read 8-bit value to used variable
(char) variable = SRAM_parameter_table.(char) variable;

//read 16-bit value to used variable
//read first higher byte to variable
(int) variable = SRAM_parameter_table.(char) variable_high;

//transfer variable bits 8 times to left
               car_door_closed <<= 8;

//read lower byte to variable with OR operation that we don't lose
// higher byte
(int) variable = ( (int) variable |
                     SRAM_parameter_table.(char)variable_low);
```

Kuva 5.5.1. Parametrien lukeminen käyttöön

5.6 Ohjelmiston ensikäynnistys ja kalibrointi

Laitteen ensikäynnistyksessä tehdään momenttikalibrointi jossa määritetään momentin nollakohta ja sen minimi- sekä maksimiarvot.

Momentin nollakohta määritetään siten. että ohjainkortti asettaa momenttiohjeen oletettuun 0-kohtaan joka vastaa 12mA:n arvona 8-bittisen muuttujan arvoa 127. Tämän jälkeen käyttäjää pyydetään säätämään arvo lisäämällä tai vähentämällä muuttujan arvoa manuaalisesti siten, että ulkoisen taajuusmuuttajan valvontanäytöllä momenttiohjeen arvo osoittaa 0%. Kun tämä arvo on säädetty kohdalleen, tallennetaan se EEPROM-muistiin offset-arvona -128..127. Tämän jälkeen momentin nollakohta lasketaan aina kaavan 5.6.1 mukaisesti:

$$torque\_zero = torque\_zero\_default + torque\_zero\_offset \quad [5.6.1]$$

Näin ollen momentin nolla-arvoksi saadaan esimerkiksi offset-arvolla -1 laskien edellisen kaavan mukaan 126.

Minimi-arvo määräytyy sen mukaan milloin ovioperaattorin auki-suuntaan ajama ovimoottori pysähtyy täysin ja luotettavasti vastavoimamoottorin momentin toimesta.

Maksimi-arvo taas toisinpäin, eli ovioperaattori käsketään manuaalisesti ajamaan ovea kiinni-suuntaan ja manuaalisesti momenttia säätämällä määritetään ovimoottorin pysähtymiseen tarvittava momentti.

Minimi ja maksimi-arvot tallennetaan EEPROM-muistiin myös arvoina -128..127 ja tämän jälkeen minimi- ja maksimimomenttien tarkistus ja laskenta tapahtuu samalla tavalla kuin nolla-kohdan arvon laskenta.

Kun ensikäynnistyksen yhteydessä tehtävä kalibrointi on suoritettu, tallennetaan EEPROM-muistin kohtaan 0x002 arvo 0x0F, joka toimii calibration_status-sanana. Tämän tila-sanan bittien tilasta tunnistetaan myös onko ensikäynnistys jo suoritettu. Jos ensikäynnistys on suoritettu, on config_ok-bitti asettuneena. Lisäksi bwd_ok, fwd_ok sekä zero –bittien tulee olla asettuneena, sillä ne kertovat momenttikalibroinnin onnistumisesta. Kalibroinnin tila-sanan bittijärjestys esitetään kuvassa 5.6.1.

x	x	x	x	config_ok	zero_ok	bwd_ok	fwd_ok

config_ok	asettuu kun ensikäynnistyskalibrointi on suoritettu onnistuneesti
zero_ok	asettuu kun momenttiohjeen nollakohta on asetettu
bwd_ok	asettuu kun momenttiraja ovi-auki suunnan vastamomentin maksimi on asetettu
fwd_ok	asettuu kun momenttiraja ovi-kiinni suunnan vastamomentin maksimi on asetettu

Kuva 5.6.1. Kalibroinnin tilasana

5.7 Laajennetun I/O-väylän alustus ja käyttäminen

Laajennetun I/O-väylän hallintaan on ohjelmistoon 3 eri funktiota. Ensimmäiseksi I/O-väylän lukemisessa tai kirjoittamisessa tarvitaan piirinvalinta-signaaleiden muodostaminen joka on esitetty kuvassa 5.7.1. Piirinvalintaan on käytetty oheispiiriä 74HC138, jonka lähdöistä 0-7 yksi asettuu sen mukaan mikä binäärinumero-arvo sen sisääntulolinjoihin on asetettu. Ulkoista piiriä hallitaan lisäksi SET_CHIP-signaalilla joka on aktiivinen loogisessa 1-tilassa. Piirinvalinta funktio suoritetaan aina kirjoitus-jakson aikana kahdesti, ensimmäisellä kerralla se saa parametrinään arvon 0..7, sen mukaan mihin ulkoiseen I/O-piiriin ollaan osoittamassa, jonka jälkeen tämä piiri asetetaan aktiiviseksi ja ohjaus-signaalin SET_CHIP-tila vaihdetaan loogisesta 0-tilasta loogiseen 1-tilaan. Kun kirjoitus on tehty, käydään funktiossa uudelleen jolloin se saa parametrinään aina arvon 0, eli varmistuksen vuoksi piirinvalintasignaali osoitetaan fyysisesti kytkemättömään linjaan ja samalla nollataan SET_CHIP-signaali. SET_CHIP-signaalille tehdään aina ehdoton-tai –operaatio, jolloin sen tila kääntyy automaattisesti.

```
int set_io_chip(unsigned char set_chip)
{
// CS -pinout && command
// nothing    0 & 7;
// SET_LCD =   1
// SET_IO1 =   2
// SET_IO2 =   3
// READ_IO =   4
// SET_M =     5
// SET_DRIVE = 6

if(set_chip <= 6)
{
              SET_IO = (set_chip & IO_MASK);
              chip_select = set_chip;
              SET_CHIP;
              return 0;
}
else return 1;
}
```

Kuva 5.7.1.Piirinvalinta-signaalin muodostaminen

Piirinvalintafunktiota voidaan kutsua kuitenkin vain write_data() tai read_input_data() –funktioista. Näillä funktioilla hallitaan dataväylää.

8-bitin väylälle kirjoitettaan write_data() -funktiolla siten että se saa parametrinaan kirjoittavan piirin numeron kuten kuvassa 5.7.1. on esitetty, jonka perusteella tieto, joka on aiemmin tallennettu kyseisen piirin tilatietomuuttujaan, kirjoitetaan data-väylään, joka on mikro-ohjaimen portti D. Kuvassa 5.7.2. on esitetty write_data() -funktion toiminta.

```
int write_data(unsigned char set_chip)
{
//set correct state-varible to output first
switch(set_chip)
{

case 1:       DATA_OUT = io_lcd;
break;
case 2:       DATA_OUT = io_1;
break;
case 3:       DATA_OUT = io_2;
break;
case 5:       DATA_OUT = io_torque;
break;
case 6:       DATA_OUT = io_drive;
break;

default: break;
}
//write data to IO chip
set_io_chip(set_chip);

//release data bus
set_io_chip(_SET_IO_OFF);

return 0;
}
```

Kuva 5.7.2.write_data() -funktio

Tiedon lukeminen sarjamuotoisesti siirrettävältä input-linjalta tapahtuu puolestaan read_data_input() -funktion avulla, joka toimii piirinvalintatoiminnaltaan täysin samalla periaatteella kuin edellä esitelty write_data -funktio. Edellisestä poiketen lukufunktio ei kuitenkaan saa parametrinaan piirin numeroa vaan se osaa itse valita siirtorekisteripiirin ja asettaa sen aktiiviseksi. Kuvassa 5.7.3 on esitelty datan lukeminen ulkoisesta siirtorekisteristä muuttujaan.

```
int read_data_input()

{
//set clock pulse counter
int count=8;
//store old value of input data to prosessing
input_data_byte_temp = input_data_byte;

//load byte to shift-register
set_io_chip(_READ_IO );
set_io_chip(0);
//start to read shift-register and read all 8 bits
do
{
              //move bit ones to left in variable that we
              //get all 8 bit to get read correct to
              //input_data_byte-variable.
              input_data_byte <<= 1;
              input_data_byte = (INPUT_DATA | input_data_byte);

              IO_CLK = 1;  // generate clock-pulse rising edge.
              IO_CLK = 0;  // generate clock-pulse falling edge.

              count--;
}
while(count>0);
if(count != 0) return 1;

else
return 0;
}
```

Kuva 5.7.3.read_data_input() -funktio

5.8 Liikkeentunnistus ja nopeuden mittaaminen

Oven liikkeen tunnistus tapahtuu käyttäen ulkoista keskeytystä. Keskeytyksen aiheuttajaksi mikro-ohjaimen keskeytyslinjaan 4 (pin 6) on kytketty pulssienkooderin kanava 1. Ulkoinen keskeytys on alustettu tapahtuvaksi jokaisella signaalin laskevalla reunalla.

Keskeytyksen tapahduttua, hyppää ohjelma keskeytysvektoriin, jonka prioriteetti on suurin käytetyistä keskeytyksistä ja vertaa pulssienkooderin kanavan 2 tilaa, jonka perusteella päätellään lisätäänkö vai vähennetäänkö door_pos-muuttujan arvoa.

Jos kanavan kaksi signaalin tila on looginen 1, tulkitaan oven liikkuvan auki suuntaan ja door_pos-muuttujaa lisätään yhdellä. Jos taas kanavan 2 looginen tila on 0, tulkitaan oven olevan menossa kiinni-suuntaan ja door_pos-muuttujaa vähennetään yhdellä. Keskeytysvektori-ohjelma on esitetty kuvassa 5.8.1.

```
// External Interrupt 4 service routine serves Channel 1 encoder pulses
interrupt [EXT_INT4] void ext_int4_isr(void)
{

if(PINE.5)    // door is opening
{
door_pos++;
if(close_end == 0) door_dir = DOOR_OPENING;
}

if(~PINE.5)   // door is closing
{
door_pos--;
if(open_end == 0) door_dir = DOOR_CLOSING;
}

// make sure that door position is inside the door width
door_state_control();

}
```

Kuva 5.8.1. Keskeytysaliohjelma oven paikannukseen.

Lisäksi aina kun keskeytys tulee, tarkistetaan myös onko ovi jo kiinni tai onko se jo auki. Oven paikannus on kaikista muista funktioista riippumaton ja sitä päivitetään aina kun keskeytys saapuu, riippumatta siitä mitä muutoin ollaan tekemässä. Oven asento-tietoja ei kuitenkaan voida päivittää silloin kun mikro-ohjain kirjoittaa tietoja omaan EEPROM-muistiinsa, tämä on estetty muistin korruptoitumisen estämiseksi. Profiileihin ja muihin asetuksiin tehtäviä muutoksia ei voida siis tallentaa silloin kun ovi on liikkeessä ja simulointitila päällä.

Oven nopeus lasketaan edellä kuvatulla tavalla 20ms:n välein. Tämä toiminta on myös riippumaton muista funktioista ja suoritetaan aina kun ajastinkeskeytys on havaittu. Nopeutta ei kuitenkaan lasketa, kuten edellä on kerrottu paikannus-funktion kohdalla. Oven nopeus-laskentafunktion kutsu sijaitsee Timer1:n keskeytysvektorissa ja kutsutoiminta on esitetty kuvassa 5.8.2.

```
door_movement_counter--;

if(door_movement_counter < 1) // door position will be saved every 20ms
{                             // there is also door speed and acc
                              // calculations when position is updated
door_movement_counter=20;     // clear counter to wait next speed count update

door_position_1 = door_position_0;  // save old position
door_position_0 = door_pos;         // sava door position at this time

// if door position does not change, clear speed and acceleration result
              if(door_position_0 == door_position_1)
              {
                door_dir = DOOR_STOPPED;
                door_speed = 0;
                door_acceleration_result = 0;
//but if configuration is ok and door does not be in it's close end
//we need to count new simulation resutl
                if( configuration_ok && close_end == 0) mechanical_load();
              }
// if door position has changed, calculate all results
              else
//if door has moved and configuration is ok, we will count new speed.
                        if(configuration_ok) door_spd();
}
```

Kuva 5.8.2. Timer1 sijaitseva nopeuslaskentafunktion kutsuminen

Oven paikkatieto tallennetaan myös 20ms:n välein, eli laskennan aikana tämä tulos pitää myös muuttaa vastaamaan mm/s –aikaväliä, jotta jatkokäsittely olisi helpompaa.

Kun nopeuden laskeminen aloitetaan, vanha tulos tallennetaan erilliseen muuttujaan, jonka jälkeen lasketaan oven tila-muutos pulsseina joka kerrotaan sillä arvolla jonka ovi liikkuu yhden pulssin aikana. Tämä arvo on muuttujassa pulse_movement, muuttujan arvo vastaa liikettä yhden pulssivälin aikana mikrometreinä.

Lopullinen tulos saadaan kun oven nopeustieto suodatetaan mediaanisuodattimen kautta ja kerrotaan suhdeluvulla 0,45; joka tuottaa tulokseksi nopeus-arvon mm/s.

Mediaanisuodatin toimii siten että siellä olevat 2 viimeistä ja nykyinen mittaustulos järjestetään suuruusjärjestykseen ja valitaan näistä keskimmäinen arvo uudeksi arvoksi. Näin saadaan suodatettua nopeusmittauksesta suuret mittausvirheet pois ja oven nopeuslaskenta näin luotettavammaksi ja tästä johtuen myös loput simulointilaskennat ovat luotettavampia. Mediaanisuodin hidastaa myös hieman momenttiohjeen muutos-aikaa, koska tarvitaan yleensä kaksi nopeuden mittaustulosta, jotta momenttiohje voi muuttua.

Nopeus-laskentafunktio toteuttaa kaavan 5.8.1 mukaisen nopeuden laskemiseen tarvittavan kaavan. Kuvassa 5.8.3 on esitetty nopeuden laskentafunktio.

$$v = \frac{s}{t} \qquad [5.8.1]$$

Jossa, v = nopeus [mm/s], s = kuljettu matka [mm] ja t = kulunut aika [s]

```
int door_spd()
{
door_speed_temp = door_speed;   // save old speed value

if(!DOOR_STOPPED)
{
//            if(door_dir == DOOR_OPENING)

// movement result is given in micrometers
movement = (door_position_0 - door_position_1) * pulse_movement / 10;

//count at ever 20ms and counting from encoder pulses.
//between encoder pulses, door is moving at 0,15mm

//if door movement-result is negative, set state-contol to be DOOR_CLOSING
              if(movement < 0)
                door_dir = DOOR_CLOSING;
//if door movement-result is positive, set state-contol to be DOOR_OPENING
              else
                door_dir = DOOR_OPENING;
}
else
              movement = 0;

//if door speed is more than 30mm/s different than last calculated value
//it need to be count again. Else we use last calculated value.
if(door_speed > door_speed_temp-30 && door_speed < door_speed_temp+30)
    door_speed = door_speed_temp;

//we also need to filterate speed measuremet little bit to avoid too
//busy speed-changed.

door_speed = speed_sampling((int)movement * 0.45);  // door speen mm/s

// count door acceleration right after speed is calculated
              door_acceleration();


return 0;
}
```

Kuva 5.8.3. door_spd() -funktio

5.9 Simulointilaskenta

Simulointia varten lasketaan oven hetkellinen kiihtyvyys ja kiihtyvyyden aiheuttama massan hitausmomentti, joka vastaa ovimekaniikan aiheuttamaa rasitusta ovioperaattorille ja sen ohjaamalle moottorille.

Kiihtyvyys lasketaan kahden edellisen hetkellisen nopeusarvon erotuksesta 20 ms:n välein joten erotukseksi saatu tulos voidaan kertoa arvolla 50, jolloin saadaan tuloksena kiihtyvyysarvo joka vastaa liikettä millimetrin neliössä. Funktiosta tallentuu siis arvo joka vastaa hetkellisen kiihtyvyyden kaavaa 5.9.1. Kiihtyvyyslaskennan toteuttava funktio on esitelty kuvassa 5.9.1.

$$a = \frac{v - v_0}{t} \qquad [5.9.1]$$

Jossa a = kiihtyvyys [mm/s^2], v = viimeksi laskettu nopeus [mm/s] ja v_0 = edellinen laskettu nopeus [mm/s].

```
int door_acceleration()
{
// save old acceleration result fisrt
door_acceleration_result_temp = door_acceleration_result;

// now counting new acceleration value for door
// movement is given in micrometers

// count new value if door is not stopped
if(door_dir > DOOR_STOPPED)
door_acceleration_result = (door_speed-door_speed_temp)*50;

// acceleration result is given now as mm/s^2
// BUT
// if door is stopped, give acceleration result to be zero
else
{
              door_acceleration_result = 0;
}

//finally go to calculate the relative mechanical load which
// comes from door mechanicue.
mechanical_load();

return 0;

}
```

Kuva 5.9.1. door_acceleration() -funktio

Kun oven kiihtyvyys on laskettu, kutsutaan kiihtyvyyden laskennan jälkeen viimeistä simulointilaskentafunktiota, jossa lopullinen hitausmomentti lasketaan. Tämä laskettu hitausmomentti vastaa sitä momenttia, joka vastavoimamoottorin tulee tuottaa, jotta simulaatio olisi mahdollisimman todenmukainen verrattuna siihen fyysiseen ovimekaniikkaan, jota kulloinkin käytössä oleva simulointiprofiili edustaa.

Funktioon on yhdistetty laskennan lisäksi siihen liittyvä oven äärilaitojen simulointi, jossa voiman simulointi asetetaan vastaamaan sitä tilannetta että ovi on täysin auki tai täysin kiinni asennossa, jolloin vastamomentti täytyy asettaa vastaamaan täyttä vastavoimaa, eli ovioperaattorin ohjaama moottori täytyy pysäyttää vastamomentin avulla. Tässä käytetään hyväksi kalibroinnin yhteydessä saatuja voima-arvoja.

Jos ovi ei ole kummassakaan reunassaan, lasketaan normaali simulointitulos. Kuormituksen laskennassa käytetään profiilista riippuvia arvoja: spring_force, door_mass, door_height, door_leave_count, guide_post_friction sekä motor_axel_radius.

Funktion suoritus tuottaa tuloksena kokonaismomentin jonka mekaaniset oven osat aiheuttavat moottorille, tämä arvo on yksiköltään mmNm:ä.

Hitausmomentin laskennan jälkeen kutsutaan vielä skaalaus-funktiota, jolla momenttiohje muodostetaan, tämän funktion toiminta kuvataan seuraavassa luvussa 5.10.

Hitausmomentin laskenta-funktio on esitelty liitteenä 2.

5.10 Momenttiohjeen muodostaminen

Momenttiohjeen muodostus D/A-muuntimelle ja taajuusmuuttajalle täytyy tehdä skaalaamalla edellä kuvattu simulointitulos prosentuaaliseksi ohjeeksi, kuten aiemmin luvussa 4.6 on esitetty. D/A-muunnin on 8-bittinen, joten momenttiohjeen arvo tällöin on 0...255, joka vastaa momenttiohjetta -100%...100%.

Tämä tehdään siten, että funktio saa parametrinaan edellä lasketun mekaanisen kuormituksen arvon, jonka yksikkö on Nmm. Tämä luku täytyy jakaa luvulla 10000 jotta saadaan luvun yksiköksi momentin perusyksikkö Nm. Sen jälkeen se pitää jakaa vielä suhdeluvulla 10 Nm, joka vastaa maksimimomenttia moottoreita yhdistävän akselin ulkoreunalla. Myös muut käytetyt arvot on suhteutettu tähän akselin ulkokehällä vaikuttavaan momenttiin.

Näin ollen parametrina tuotu arvo voidaan suoraan jakaa luvulla 100000, jotta saadaan suhdeluku vastaamaan prosentuaalista arvoa. Joka tämän jälkeen suhteuttaa 8-bittisen momenttimuuttujan vastaamaan edellä laskettua kuormamomenttia.

Nyt simuloitu kuormamomentti on saatu vastaamaan arvoa joka voidaan ohjata taajuusmuuttajalle momenttiohjeena ja se kirjoitetaan simulointilaskelmafunktiosta kutsuen siten kuin luvussa 5.9 on esitetty.

Muunnos-funktio on esitelty kuvassa 5.10.1.

```
int load_conversion_to_guideline(double load_to_converse)
{
double suhdeluku;

suhdeluku = (double) load_to_converse / 100000;
// get total_load -result in mmNm and give result as Nm


temporary_load = (double)torque_zero *  suhdeluku;


return 0;
}
```

Kuva 5.10.1. load_conversion_to_guideline() -funktio

5.11 A/D-muuntimen lukeminen ja mittaustoiminnot

A/D-muuntimella luetaan mittaustiedot ovioperaattorin syöttämästä jännitteestä ja virrasta moottorijohtimiin, sekä taajuusmuuttajalta tuleva takaisinkytkentäsignaali moottorin momentista. A/D-muuntimen lukeminen tapahtuu siten kuin se on piirinvalmistajan toimesta datalehdessä ilmoitettu. Lukufunktio on esitetty kuvassa 5.11.1.

```
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
```

Kuva 5.11.1. read_adc() -funktio

A/D-muuntimen lukemisessa on otettava huomioon AVR-tuoteperheen prosessoreilla oleva sisäinen kapasitanssi (14pF), joka saattaa vaikuttaa mittaustulokseen jos mitattavan kohteen impedanssi on suurempi kuin 10k ohmia. . Piirinvalmistaja suosittelee S/H-piirin vakavoimiseksi ylimääräistä mittausta ennen varsinaista mittausta jo mitattavan kohteen impedanssi on ylempi kuin 10k ohmia /2/. Sovelluksessa käytettävä TL081 operaatiovahvistin, joka tekee mittausjännitteen jokaiseen A/D-muunnin linjaan, on impedanssiltaan maksimissaan 100 ohmia ja sarjavastukset ovat arvoltaan 1k ohmia, tästä johtuen ongelmia ei pitäisi olla ja näin ollen ylimääräistä mittauskertaa ei sovelluksessa suoriteta ajan säästämiseksi

Näytteet otetaan aina peräkkäin jokaisesta kanavasta jotta näyteikkuna olisi mahdollisimman pieni kuten edellä on todettu. Mittaustulokset tallennetaan jokaisen mitattavan kohteen osalta 10-arvoiseen mittaustulos jonoon, josta arvot kirjoitetaan järjestyksessä myös muistiin.

5.12 Sisäisen EEPROM-muistin hallinta

Mikro-ohjaimella on 2kB sisäistä EEPROM-muistia jossa säilytetään konfigurointi ja kalibrointitietoja, sekä oviprofiilien asetus-arvoja. EEPROM-muistinhallintatoiminnot on tehty siten että kaikki muut toimenpiteet näiden aikana on estetty, jotta muistin käsittely on turvallista ja korruptoitumisen mahdollisuus on näin estetty. Tämä on mahdollista estämällä ulkoiset ja sisäiset keskeytykset komennolla #asm(”cli”) ja sallia ne voidaan komennolla #asm(”sei”). Cli-komento nollaa prosessorin tila-rekisteristä keskeytys-maskibitin ja sei asettaa sen.

Ohjelmiston EEPROM-hallintaa varten on kaksi erillistä funktiota, toinen kirjoittamiseen ja toinen lukemiseen. Muistinhallinta tapahtuu siten kuin se on esitetty piirinvalmistajan datalehdessä /3/. Kuvissa 5.12.1. ja 5.12.2. on esitetty nämä funktiot.

Kirjoitus-funktio write_eeprom_byte saa parametreinään osoitteen johon kirjoitetaan, sekä kirjoitettavan datan. Luku-funktio read_eeprom_byte saa parametrikseen vain luettavan osoitteen, jonka sisällön se lopuksi palauttaa.

```
int write_eeprom_byte(unsigned int eeprom_address, unsigned char data)
{
// Global enable interrupts
#asm("cli")
  while(EECR & (1 << EEWE));       //wait for completion of previous write cycle

  EEAR = eeprom_address;
  EEDR = data;

  //set EEPROM to write mode
  EECR |= (1 << EEMWE);
  //Start writing to EEPROM
  EECR |= (1 << EEWE);
#asm("sei")
return 0;
}
```

Kuva 5.12.1. write_eeprom_byte() -funktio

```
unsigned char read_eeprom_byte(unsigned int eeprom_address)
{
#asm("cli")
  while(EECR & (1 << EEWE));  //wait for completion of previous write cycle

  EEAR = eeprom_address;

  //Start reading from EEPROM by writing EERE bit to EECR register
  EECR |= (1 << EERE);
#asm("sei")
return EEDR;
}
```

Kuva 5.12.2. read_eeprom_byte() –funktio

5.13 Muistikortin hallinta

Muistikortin hallintaan on käytetty valmista GNU –lisenssillä olevaa ohjelmistoa ja sen funktioita. Tämä ohjelmisto sisältää suoraan FAT16/32 yhteensopivan tiedostonhallintajärjestelmän.

Ohjelmiston ja sen sisältämien funktioiden kuvaus löytyy ajurin ohjekirjasta ja lähdekoodista /11/.

5.14 Käyttöliittymä ja sen hallinta

Paikallisen käyttöliittymän hallintaan tarvittavat funktiot liittyvät näppäinten lukuun, joka on jo kuvattu luvussa 5.7. sekä lisäksi LCD-näytön hallintaan rakennetut funktiot sen alustamista ja ohjaamista varten.

Näppäinten luku tapahtuu siis edellä kuvatulla tavalla 10ms:n välein, pääohjelmassa tutkitaan näppäinten tilavaihdokset samalla aika-välillä käyttäen hyväksi Timer0:n luomaa tila-signaalia hyväksi käyttäen. Näppäinten painallus tuottaa aktiivisen toiminnan sen laskevalla reunalla. Eli, näppäimiä tutkitaan siten että jos edellinen tila oli looginen 1 tila ja

viimeksi haettu arvo on looginen 0, on tämä laskeva reuna ja tilamuutos tuottaa toiminnan riippuen sen hetkisestä muusta toimintatilasta. Näppäinten lukuun käytetään input_data_byte ja input_data_byte_temp -muuttujia.

LCD –näytön ohjauksen ajastuksessa käytetään Timer0:n muodostamaa keskeytys-signaalia jolla hallitaan muiden toiminteiden lisäksi lcd_timer-muuttujaa. Näytön ohjauksessa käytetyt viiveet ja muut ajastukset on tehty kuten valmistajan datalehdessä on kerrottu /8/.

LCD-näytölle voidaan tulostaa joko yksittäisiä merkkejä write_lcd_data() -funktion avulla tai merkkijonoja write_str() -funktion avulla. Koska LCD-näytön hallinta-signaalit E ja RS ovat eri IO-piirin ohjauksessa kuin dataväylä, ohjelmistossa on näille signaaleille myös omat funktionsa, joita kuitenkin hallitaan vain näytön alustus ja kirjoitus-funktioista. LCD-näytölle voidaan kirjoittaa myös tiettyyn kohtaan näyttöä lcd_solu() -funktion avulla, jonka avulla voidaan osoittaa mitä tahansa näytön solua.

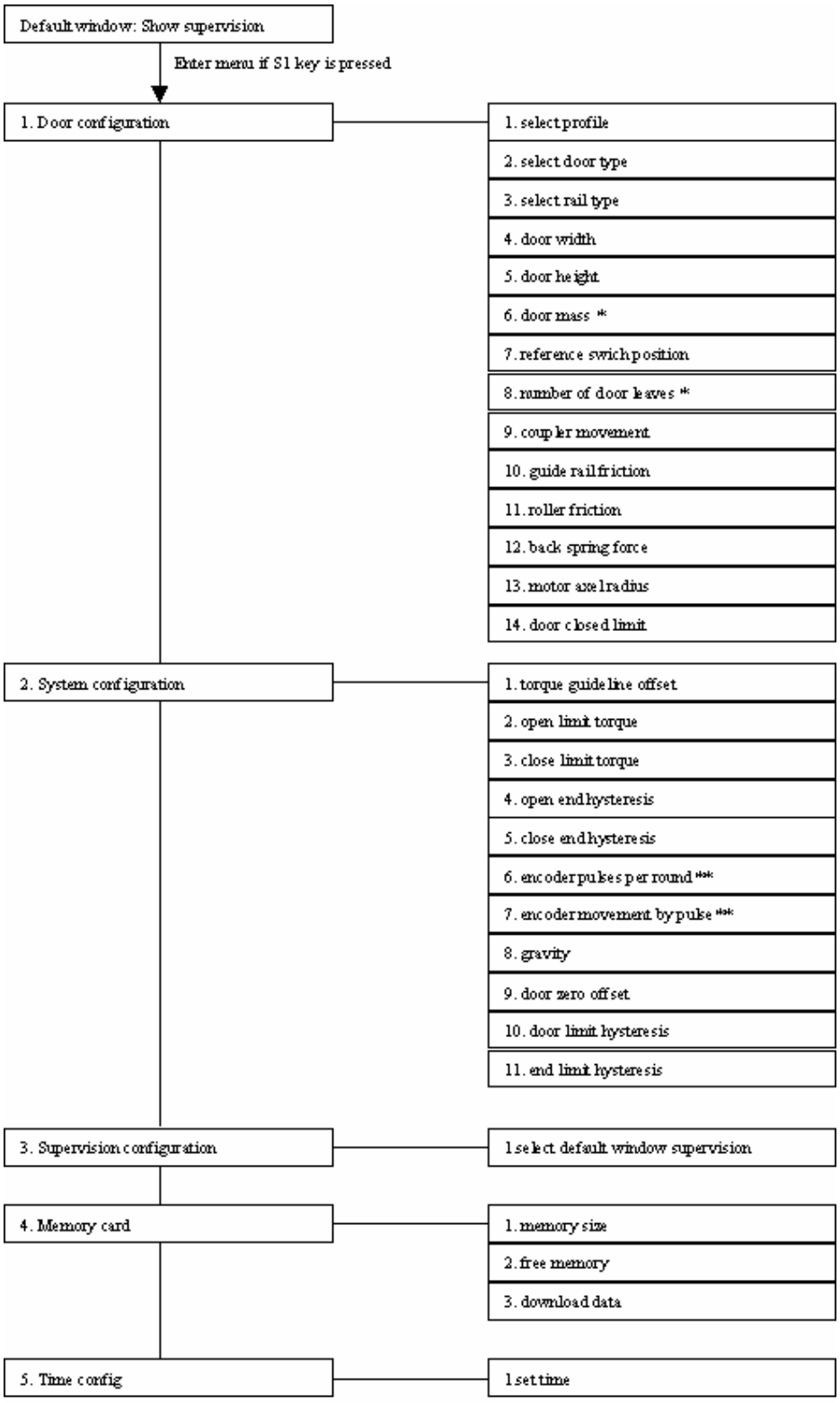
LCD-näytön ohjaus-funktiot on kuvattu kuvissa 5.14.1 ja 5.14.2 sekä paikallisen käyttöliittymän hallintarakenne kuvassa 5.14.3.

```
void lcd_init()
{
set_lcd_e(0);
write_lcd_data(0x38,0);
lcd_delay(100);                         //wait 10ms
write_lcd_data(0x38,0);    //set to 8bit interface
lcd_delay(20);                    //wait 200us
write_lcd_data(0x38,0);       //set to 8bit interface
lcd_delay(2);
write_lcd_data(0x06,0);
lcd_delay(2);
write_lcd_data(0x0E,0);
lcd_delay(2);
write_lcd_data(0x01,0);               // set display on after initializion
lcd_delay(20);
lcd_handler = (lcd_handler | 0x01);   // set lcd initialized bit in register
}
void set_lcd_e(unsigned char state)
{
if(state == 0) LCD_E_OFF;
if(state == 1) LCD_E_ON;
//write data to IO chip
write_data(_IO2);
}

void set_lcd_rs(unsigned char state)
{
if(state == 0) LCD_RS_OFF;
if(state == 1) LCD_RS_ON;
//write data to IO chip
write_data(_IO2);
}
```

Kuva 5.14.1. LCD ohjaus-signaalien hallinta –funktiot

```
void write_str(unsigned char flash *str)
{
int x;
int y;
y=strlenf(str);                       //testataan merkkijonon pituus!
for(x=0; x<y; x++)
{
write_lcd_data(str[x],1);
}
}

void write_lcd_data(unsigned char data, unsigned char RS_select)
{
io_lcd = 0;
if(RS_select == 0) LCD_RS_OFF;
if(RS_select == 1) LCD_RS_ON;
LCD_E_ON;
write_data(_IO2);
             io_lcd = data;
             write_data(_LCD);

lcd_delay(1);
LCD_E_OFF;
write_data(_IO2);

// does it need to change also RS-pin?
if(RS_select == 0) LCD_RS_ON;
if(RS_select == 1) LCD_RS_OFF;

}
```

Kuva 5.14.2 LCD kirjoitus –funktiot



Kuva 5.14.3 Paikallisen käyttöliittymän hallintarakenne

5.15 Pääohjelma

Pääohjelmassa hallitaan aliohjelmien toteutusjärjestystä ja toteutetaan ei-kriittisiä toimintoja keskeytysaliohjelmien välissä, sekä tehdään näiden ohjaamina tila-sanojen vaatimat toimenpiteet.

Mikro-ohjaimen käynnistyttyä pääohjelma suorittaa ensin edellä kuvatut alustustoimenpiteet, lukee EEPROM –muistista tarvittavat tiedot, jonka jälkeen ladataan muistista myös käytössä oleva oviprofiili.

Kun alustustoimenpiteet ohjaimelle ja kaikille muuttujille on tehty, pääohjelma siirtyy pääohjelmasilmukkaan, joka on tyypiltään päättymätön ja näin ollen pääohjelma toimii kokoaikaisesti yhdessä silmukassa eikä poistu siitä kuin aliohjelmakutsujen ajaksi. Kun aliohjelma on suoritettu, palataan suorittamaan pääohjelmasilmukkaa ja tehdään seuraavana vuorossa olevat toimenpiteet tila-sanoista riippuen.

6 TESTAUS

6.1 Testausmenetelmät

Ohjainkortin testauksessa on pyritty keskittymään elektroniikan ja ohjelmiston luotettavuuteen, koska kyseessä on testilaite. Laitteiston tulee näin ollen pystyä toistamaan luotettavasti sama toiminta kun sen parametrointi pysyy vakiona.

Ohjainkortin elektroniikan testaus alkoi ennen ohjelmiston kehitysversioiden lataamista kortille siten että oskilloskooppia ja yleismittaria käyttäen todettiin kortin eri pisteiden jännite-arvojen paikkaansa pitävyys suunniteltuun nähden.

Kun mittaukset oli tehty, ohjelmoitiin kortille testaus-ohjelmisto, jonka avulla todettiin kaikkien loogisten operaatioiden ja prosessorin ohjaamien analogia-signaalien toiminta ja niiden vastaavuus suunnittelun mukaisiin arvoihin.

Testaus-vaiheessa löytyneitä vikoja korjattiin aina löydettäessä. Tällaisia olivat muun muassa ensimmäisen version riittämätön tehonsyötön jäähdytys, piirinvalintasignaalien toiminnan varmistaminen erillisellä sallinta-signaalilla, momenttiohje-lähdön riittämätön virransyöttökyky sekä joidenkin signaalien suodatustarve.

Suodatusta vaativiin signaaleihin, kuten momenttiohjeen lähtöön lisättiin suodatus-kondensaattori, jonka jälkeen signaalin häiriöt poistuivat.

Tehonsyötön jäähdytystä lisättiin erillisellä jäähdytyslevyllä. Momenttiohjesyötön lähtöön lisättiin lisäksi transistori-puskuri, jotta virransyöttö olisi riittävä ja kattaa koko virtaviestin alueen 4-20mA.

Erillinen sallinta-signaali lisättiin prosessorin I/O-nastan ja piirinvalintasignaalin U4 väliin, jotta piirinvalinta-signaalin vaihtuminen toiseen ei tuottaisi häiriöitä muiden I/O-piirien toimintaan.

Ohjelmiston jatkokehitystä jatkettiin, kun ohjainkortin elektroniikka oli todettu toimivaksi ja näin ollen ohjelmiston testaamisen kannalta edulliseksi. Ohjelmiston testaus on tehty jokaiselle ohjelmiston toiminnalliselle funktiolle ja toiminnekokonaisuudelle erikseen aina kun ohjelmisto on tältä osin saatu valmiiksi. Saman aikainen testaus prosessorin simulointiohjelmistolla ja varsinaisessa toiminnallisessa ympäristössä, eli ohjauskortilla on osoittautunut edulliseksi ja nopeaksi tavaksi saada ohjelmalliset toiminteet ja toimintakokonaisuudet toimiviksi ennen kuin seuraavaa ohjelmointivaihetta on aloitettu.

Testaus-laitteiston toiminnallinen testaus on tehty kokoaikaisesti siten että siinä on ollut kiinni oikea ovioperaattori jota on ohjattu manuaalisesti. Ohjelmistokehityksen ollessa siinä

vaiheessa että laite on todettu toimivaksi manuaali-ohjauksessa, voidaan sanoa että tässä tutkintotyössä kuvattu simulointilaitteiston ohjainkortti on testattu täysin.

Tämän jälkeen simulointilaitteistoa testattiin myös ovioperaattorin varsinaisessa ympäristössä, eli osana hissin ohjausjärjestelmää. Oikeassa toimintaympäristössä, jossa laitteen ohjelmiston ja elektroniikan toimintaa testattiin mittauksin ja sen toimivuus todettiin käyttäen pitkiä testausjaksoja ja todennusmittauksia sen suhteen, kykeneekö hissi toimimaan kuten oikean ovikoneiston kanssa. Ympäristötestauksen dokumentointi jää vain KONE Oyj:n käyttöön.

Ohjainkortin toiminnallisessa testauksessa käytettiin signaalien toimivuuden ja oikean loogisen toiminnan todentamiseen oskilloskooppia sekä logiikka-analysaattoria.

6.2 Tulokset

Elektroniikan toimintamittaukset

Testauspisteet sekä kortin liittimien nastat mitattu ja todettu että jännitearvot ovat kohdallaan. Testauspisteet löytyvät ohjainkortin kytkentäkaaviosta /1/.

Testitulos: Läpäisty

Testaus-ohjelmiston avulla suoritetut toiminnalliset mittaukset

Testausohjelmistolla testattiin kaikki kortilla olevat I/O-piirit ja liitäntöjen toimivuus.

Testitulos: Läpäisty

Manuaalinen toimintatestaus

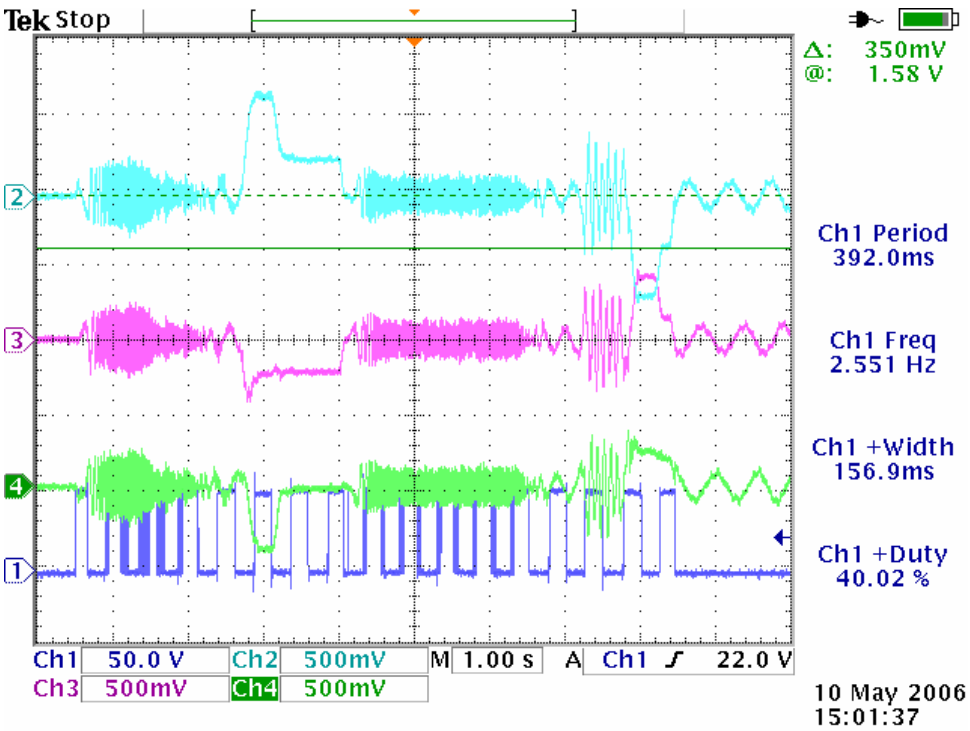
Manuaalinen toiminta-testaus on suoritettu ohjelmistosuunnittelun edetessä toiminnallisin kokonaisuuksin. Toiminnallisten kokonaisuuksien testauksen jälkeen tehtiin vielä koko ohjelmiston toiminnasta.

Testauksen aikana ovioperaattoria ohjattiin manuaalisesti auki- ja kiinni –suuntiin, jonka aikana tietoa tallennettiin niin ohjainkortilta sarjaliikenneväylän kautta kuin mittauksin ohjainkortin momenttiohjeesta. Lisäksi tutkittiin ovimoottorin virtojen vastaavuutta momentin nousuun ja laskuun.

Testitulos: Läpäisty

Lopuksi mittaukset tehtiin koko laitteiston osalta tarkistellen moottorivirtoja ovimoottorissa, joka on simuloidun kuorman kuormituksen kohteena.

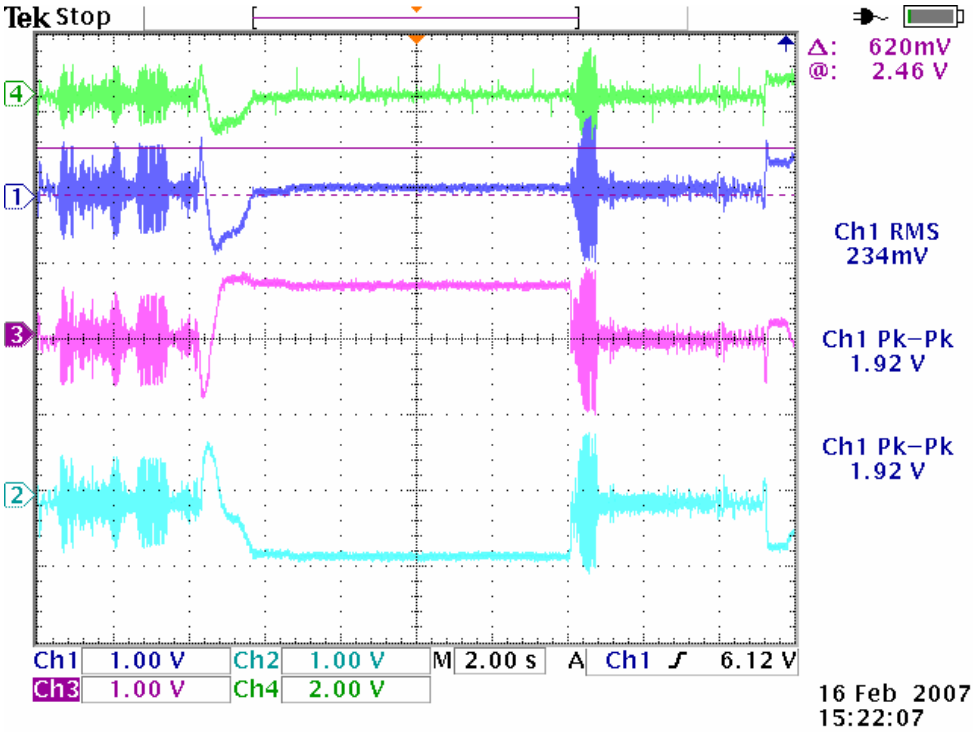
Kuvassa 6.2.1 on esitetty virranmittaus työn määrittelyvaiheessa.



Kuva 6.2.1 Virran mittaus ovimoottorista ennen työn aloittamista

Kuvasta 6.2.1. selviää ovimoottorin tarvitsema virta. oskilloskoopin kanavat 2-4. Virran arvo kiihdytys-kohdissa on noin 2-2,5 ampeeria. Tasaisen virran kohta on se kohta jossa ovi on kokonaan auki. Tässä mittauksessa ovioperaattorin ohjauksessa oli noin 35 kilogramman kuorma. Kanavassa 1 oleva signaali ei liity virtojen mittaukseen.

Kuvassa 6.2.2. on vertausmittausten suorituksesta samanlainen moottorivirtojen mittauskuva. Kuvasta nähdään että vastaavan kiihdytyskohdan virta kasvaa tässä noin 6,5 ampeeriin. Vertaus on tehty kuvan 6.2.1. ajan hetkellä 1 sekunti ja kuvan 6.2.2 ajan hetkellä noin 1 sekunti, jotka vastaavat toisiaan. Vertailumittauksessa simulointi oli asetettu 600kg:n kuormitus. Tästä voidaan todeta että simuloitu kuormamomentti vaikuttaa oikein ovioperaattorin ohjausvirtaan ja näin ollen ovimoottorin momenttiin. Kuvan 6.2.2. kanavassa 4 näkyvä signaali kuvaa kortin mittaustietoa.



Kuva 6.2.2. Virran mittaus ovimoottorista simulaattorilaitteistossa

7 Päätelmät ja yhteenveto

Työn haasteellisuus oli huomattavan suuri, johtuen siitä että laitteistolla tuli kyetä simuloimaan poikittais-liikkeisen massan hitausmomentin lisäksi myös oven mekaanisia rajoja. Tällöin voiman tuoton muutokset ja tarkkuudet astuivat vahvasti häiritsemään varsinaista hitausmomentin simulointia. Erityisen haasteelliseksi ohjausta suunniteltaessa osoittautui ovimoottorin hallittu pysäyttäminen tiettyyn raja-kohtaan siten että vapaasti pyörivät moottoreiden roottorit eivät liiku.

Ohjelmistosuunnittelun haasteet liittyivät niin ikään momentti-ohjauksen ja paikkatiedon laskentaan ja niiden nopeaan muuttamiseen. Lisäksi projektin alussa käytetystä Codevision AVR ohjelmistonkehitysympäristöstä piti luopua pienten yhteensopivuusongelmien vuoksi ja tilalle valittiin vapaalla lisenssillä käytettävä WinAVR – kääntäjäympäristö, jonka johdosta kahdenvälisiä yhteensopivuusongelmia löytyi yllättävän paljon. Ongelmallisiin kääntäjien välisiin eroihin löytyi kuitenkin ratkaisu ja ohjelmiston kehitystä voitiin jatkaa uudessa ympäristössä entistä tehokkaammin ja selkeämmin.

Kehitys-ongelmia lisäsi myös hienoisesti ovioperaattorin ohjaus ja paikannustapa sekä hissin ovirakenteen mekaaninen toteutustapa.

Simulointilaitteiston teknistä dokumentointia on suoritettu yhdessä tämän tutkintotyön kanssa ja se julkaistaan KONE Oyj:n käyttöön sen valmistuttua.

Spesifioidun testauslaitteen kaupalliset hyödyt ovat välillisiä ja niiden hyödyntäminen tapahtuu laitteen käyttötarkoituksen mukaan. Näin ollen kaupallisia tarkoituskohteita ei voida eritellä tässä.

Tämä tutkintotyö on kuvaus kokonaisen laitteiston osasta ja projektin jatkuminen ohjauskortin valmistuttua saattaa muuttaa tässä kuvattua ohjelmistoa jossain määrin. Laitteiston jatkokäyttö edellyttäisi kuitenkin myös hienoisia muutoksia itse ohjauskorttiin, riippuen käyttötarkoituksesta tai uusista testauskohteista ja niiden ohjauksesta.

Tämä ohjainkortti on suunniteltu määrittelyn mukaisille ovioperaattorityypeille ja niissä käytettyjen ohjausten mukaisesti.

LÄHDELUETTELO

1. Ohjainkortin kytkentäkaavio

2. VACON –taajuusmuuttajan tekniset tiedot BC00199A [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: http://www.vacon.com/Default.aspx?id=463408

3. VACON –taajuusmuuttajan käyttöohje UD1029A [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: http://www.vacon.com/Default.aspx?id=462999

4. Atmel Corporation, ATMega64 datalehti [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: http://www.atmel.com/dyn/resources/prod_documents/doc2490.pdf

5. Clare an IXYS Company, LCA110L datalehti [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: http://www.clare.com/home/pdfs.nsf/www/LCA110L_R1_0.qxd.pdf/$file/LCA110L_R1_0.qxd.pdf

6. Analog Devices, AD7524 datalehti [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: http://www.analog.com/UploadedFiles/Data_Sheets/AD7524.pdf

7. Fairchild Semiconductor, HCPL-0611 datalehti [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: www.fairchildsemi.com/ds/HC/HCPL-0601.pdf

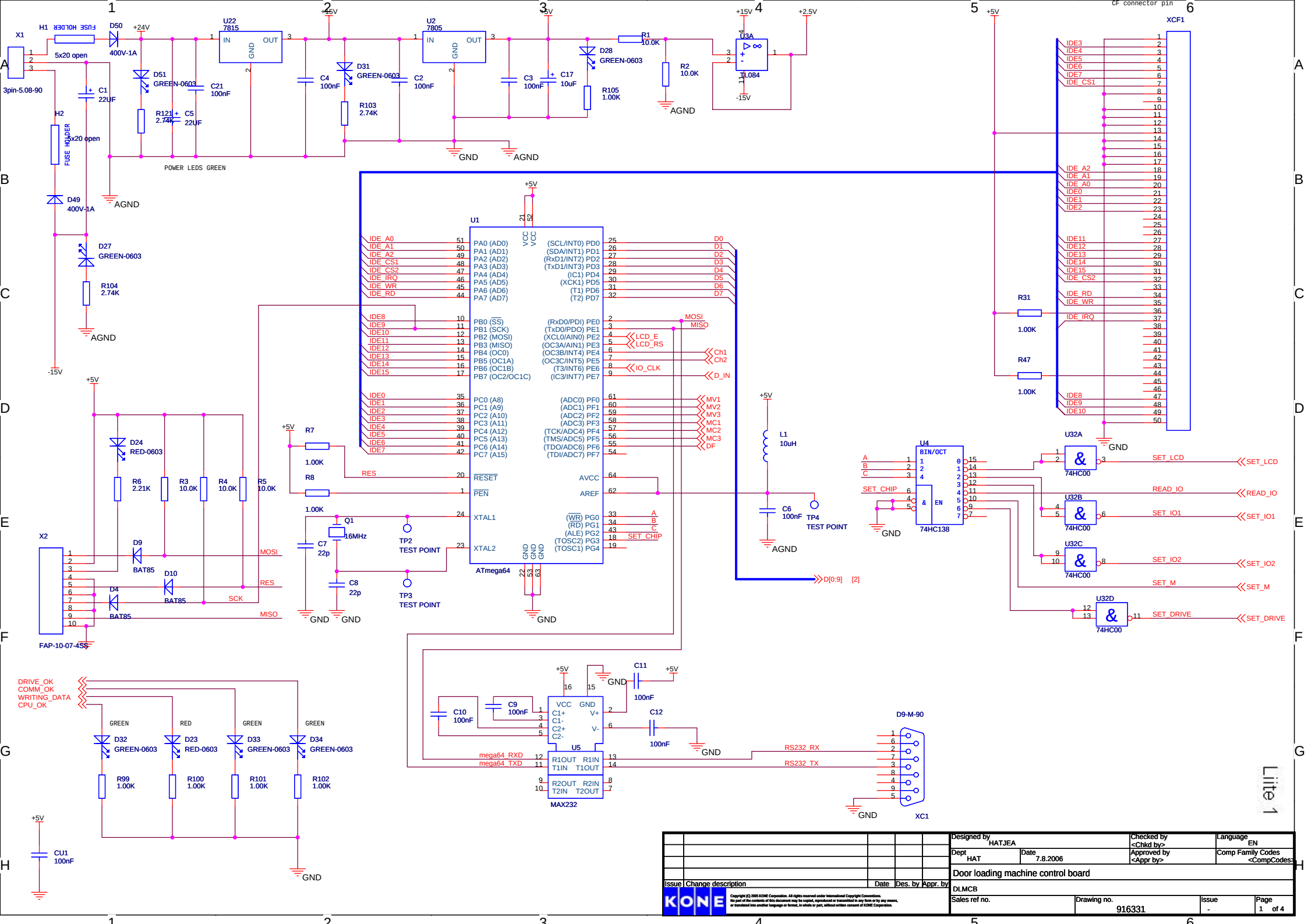
8. HCPL-0611 korvaavien komponenttien luotettavuusmittaukset [(KONE Oyj R&D laboratory –J.Angervuo)

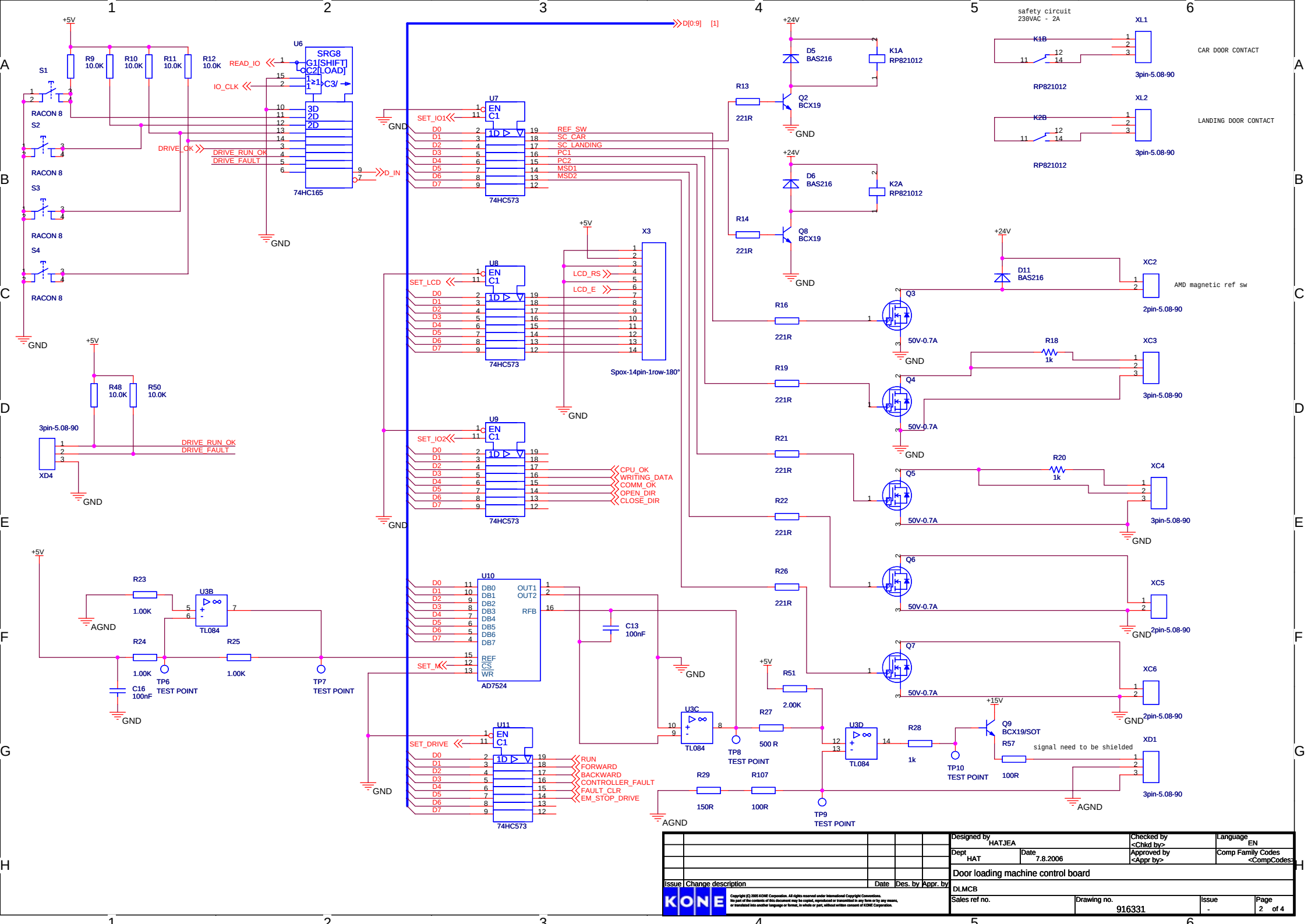
9. Seiko, L2032-B1J datalehti [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: http://www.elfa.se/pdf/75/07555089.pdf

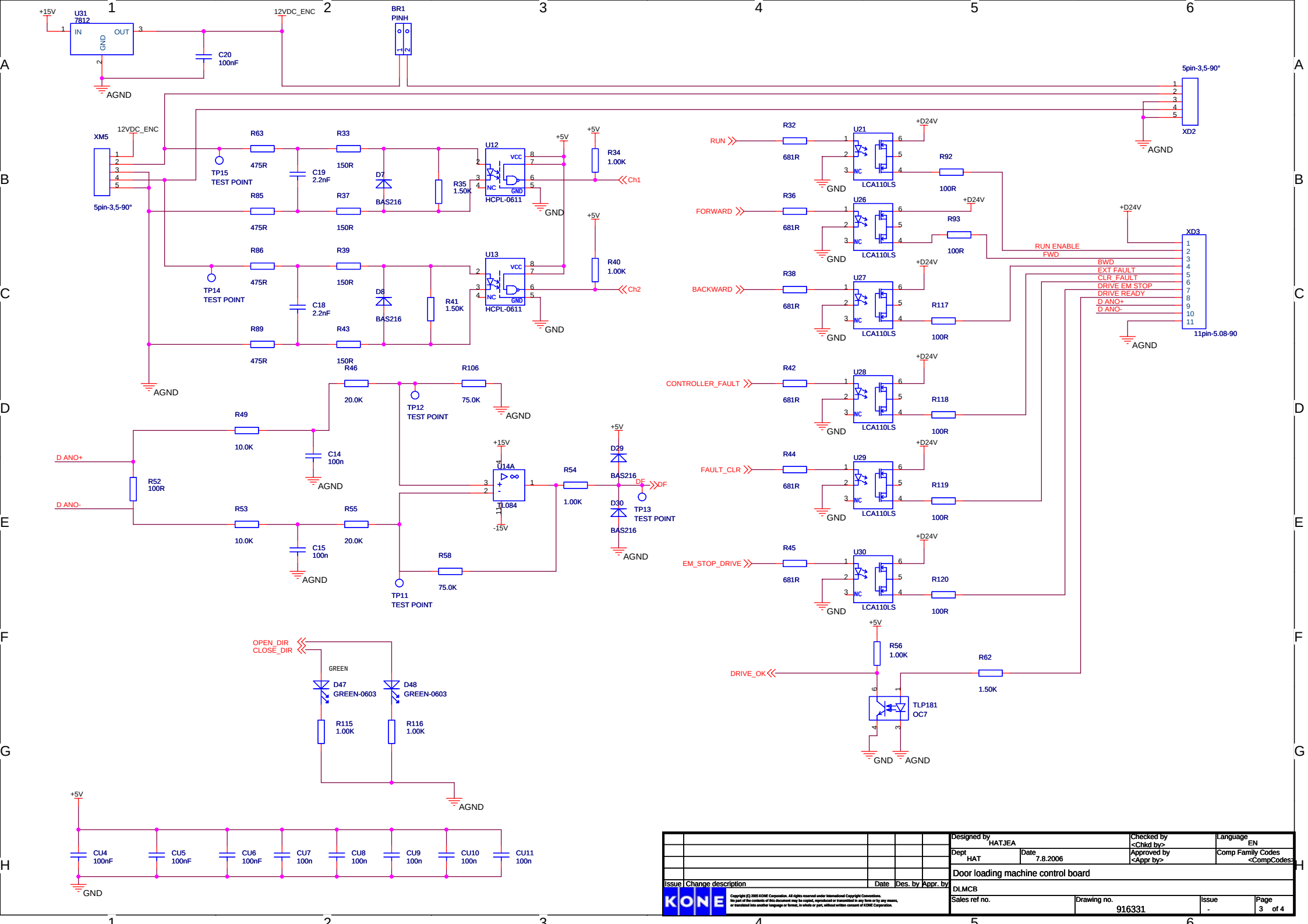
10. VAC Vacuumschmelze, N4644-X400 datalehti [sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: http://www.vacuumschmelze.de/dynamic/docroot/medialib/documents/pdf/kbproduktblaetter/4644-X400.pdf

11. Angelo Bannack, Giordano Bruno Wolaniuk, FAT16-32 File System Driver for ATMEL AVR,[sähköinen dokumentti] [viitattu 14.2.2007] saatavissa: Saatavissa: http://www.e-armazem.com.br/dev/fat16-32driver/

LIITELUETTELO

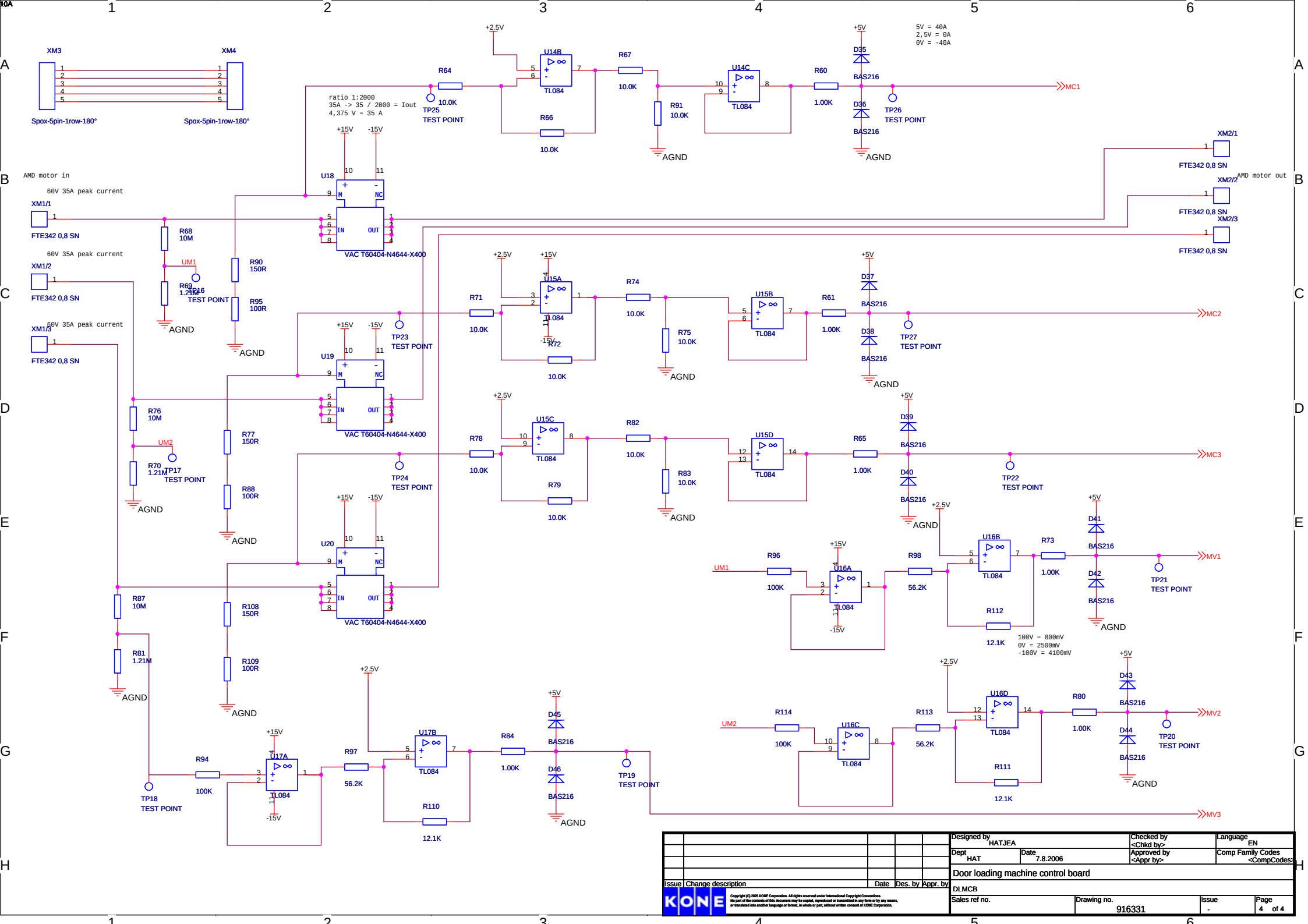



safety circuit
230VAC - 2A
CAR DOOR CONTACT
LANDING DOOR CONTACT
RP821012
3pin-5.08-90
2pin-5.08-90
AMD magnetic ref sw
Spox-14pin-1row-180°
74HC165
74HC573
AD7524
TL084
TEST POINT
RACON 8
50V-0.7A
BCX19/SOT
BCX19
BAS216
signal need to be shielded
D[0:9] [1]
Designed by HATJEA
Checked by <Chkd by>
Language EN
Dept HAT
Date 7.8.2006
Approved by <Appr by>
Comp Family Codes <CompCodes>
Door loading machine control board
DLMCB
Sales ref no.
Drawing no. 916331
Issue -
Page 2 of 4
Issue Change description Date Des. by Appr. by



Issue	Change description	Date	Des. by	Appr. by

Designed by HATJEA		Checked by <Chkd by>	Language EN
Dept HAT	Date 7.8.2006	Approved by <Appr by>	Comp Family Codes <CompCodes>
Door loading machine control board			
DLMCB			
Sales ref no.	Drawing no. 916331	Issue -	Page 3 of 4



Issue	Change description	Date	Des. by	Appr. by

Designed by HATJEA		Checked by <Chkd by>	Language EN
Dept HAT	Date 7.8.2006	Approved by <Appr by>	Comp Family Codes <CompCodes>

Door loading machine control board

DLMCB

Sales ref no.	Drawing no.	Issue	Page
	916331	-	4 of 4

Toiminnallinen määrittely

DLMCB – Controller software

Dokumentin numero	916339H01
Versio	1.0
Päiväys	16.2.2006 – 10.48
Tulostettu	3/27/2007 12:23
Tekijä	Jani Angervuo
Omistaja	Jani Angervuo
Hyväksyjä	Jani Angervuo

VERSIOHISTORIA

Versio	Päiväys	Tekijät	Selite
0.1	22.8.2006	Angervuo	Alkuperäinen
1.0	16.2.2007	Angervuo	Versiomuutos - funktiokuvauksen poistettu (esitetään lähdekoodissa) Käyttöversio

SISÄLLYSLUETTELO

1. JOHDANTO

1.1 Tarkoitus ja kattavuus

Tämä dokumentti on tarkoitettu tutkintotyön työohjeeksi ja sen sisältämän ohjaus-ohjelmiston kuvaukseen niin ohjelmiston kehitys- kuin ylläpitovaiheessa.

Dokumentti kattaa järjestelmän ohjelmiston suunnittelun ja toimintavaatimukset kokonaisuudessaan.

1.2 Tuote

Tämä dokumentti kattaa AMD1 ja AMD2 ovityypeille tarkoitetun kuormitus-testauslaitteen ohjaus-ohjelmiston määrittelyn.

Ohjelmisto toimii 8 bittisessä RISC-mikro-ohjain –ympäristössä.

1.3 Määritelmät, termit ja lyhenteet

Termi	Määritelmä
RISC	Mikroprosessori-tyyppi joka suorittaa yhden käskyn yhden konejakson aikana (Reduced Instruction Set Computer)
A/D-muunnin	Muuttaa analogisen signaalin digitaaliseen muotoon
D/A-muunnin	Muuttaa digitaalisen arvon analogiseksi signaaliksi
RS232	Sarjaliikenneprotokolla
AMDx	Hissin ovioperaattorin tyyppi (x=1; 1,5; 2)
LCD	Nestekidenäyttö
AVR	Atmelin AVR-mikro-ohjaintuoteperhe
I/O -valitsin	Sovelluksessa käytettävä oheispiiri, jonka avulla käytetään laajennettua I/O-väylää
Valinnainen tallennustieto	Käyttäjä saa valita tallennetaanko ko. tieto muistiin testauksen yhteydessä
CF	CompactFlash, muistikorttityyppi.
Stand-alone tila	Sovelluksen yksi tila, jossa simuloidaan ovioperaattorin toimintaa itsenäisesti.

1.4 Yleiskatsaus dokumenttiin

Luku 2 kuvaa järjestelmän toiminnan yleisellä tasolla: Siihen kuuluvan laitteiston, käyttäjät, järjestelmän riippuvuudet ja rajoitukset.

Luvussa 3 kuvataan järjestelmän tietosisältö.

Luvussa 4 määritellään järjestelmän toiminnot.

Luvussa 5 määritellään järjestelmän ulkoiset liittymät, eli laitteiston, tietoliikenteen ja ohjelmistoliittymät.

Lukuun 6 on kirjattu järjestelmän ei-toiminnalliset ominaisuudet, kuten suorituskyky, vasteajat, käytettävyys ja ylläpidettävyys.

Luvussa 7 on kirjattu ohjelmiston suunnittelun rajoitteet, kuten standardit ja ohjelmistosta sekä laitteistosta johtuvat rajoitteet.

2. YLEISKUVAUS

2.1 Ympäristö

Järjestelmä toimii 8-bittisessä RISC-mikro-ohjainympäristössä. Ohjelmisto ladataan mikro-ohjaimen vapaasti ohjelmoitavaan flash-muistiin.

Ohjelmiston tehtävänä on ohjata ovioperaattorin toimintaa, siten kuin se toimii reaalisesti sekä vastavuoroisesti vastavoimamoottorin toimintaa siten kuin ovimekaanikan mekaaniset osat vaikuttavat reaalisesti. Vaihtoehtoisesti ohjelmisto ohjaa vain vastavoimamoottoria sen perusteella, miten ovi-operaattori toimii itsenäisesti.

Ohjelmisto kerää ajo-tiedot ja tallentaa ne CompactFlash –muistikortille myöhempää käyttöä varten.

Ohjelmistossa on lisäksi sarjaliikenneohjaus, jonka avulla parametrit voidaan lukea ja järjestelmän toimintaa seurata.

2.2 Toiminta

Ohjelmisto alustaa mikro-ohjaimen ja siihen liittyvät oheis-komponentit. Alustuksen yhteydessä alustetaan tarvittavat tulo- ja lähtöportit, ajastimet sekä sarjaliikenneväylä ja A/D-muunnin.

Alustuksen yhteydestä muistista luetaan myös tallennetut parametroinnit. Muisti-aluekartta löytyy luvusta 3.

Alustus-rutiinien jälkeen ohjelmisto jää valmius-tilaan, jossa ohjelmiston parametrointi, tiedonkeruu ja muut asetukset sekä käynnistys –käskyt voidaan.

Käynnistyksen jälkeen laitteisto aloittaa testaus-rutiinit annettujen parametrien mukaan.

Laite tallentaa vain mittaustiedot myöhempää analysointia varten.

Dokumentin liitteenä on ohjelmiston tila-kaavio (Liite 1)

2.3 Käyttäjät

Järjestelmää käytetään tutkimus- ja tuotekehitys sekä tuotantotestaus-tarkoituksiin. Järjestelmä on yhden käyttäjän järjestelmä.

Laitteiston parametroinnin yhteydessä voidaan käyttää PC-tietokoneelta löytyvää sarjaliikennepäätettä tai parametrointi voidaan tehdä manuaalisesti laitteen omasta käyttöliittymästä.

Laitteen parametroinnin apuna voidaan käyttää erillistä parametrilistaa sekä laitteen toiminnallista selostusta.

Laitteen käyttäjä tarvitsee perustiedot laitteen käyttämisestä. Laitteen käyttäjän ei tarvitse kuin asentaa testattava kortti paikalleen ja käynnistää laite.

2.4 Oletukset ja rajoitteet

Laitteen mekaaninen ja sähköinen versionumero määrittää kummalle korttityypille laitteistoa käytetään.

Ohjelmisto on yhteensopiva molempien ovi-operaattorikorttien kanssa.

Laitteen oma käyttöliittymä toimii neljällä (4) painikekytkimellä ja 2x20 –merkkisellä LCD-näytöllä.

Sarjaliikenne-käyttöliittymään ja muuhun sarjaliikenteeseen käytetään seuraavia asetuksia: 9600 bit/s, 8 data-bittiä, ei pariteettia, 1 lopetus-bitti. Kättelytoiminto ei ole käytössä.

3. TIEDOT JA TIETOKANNAT

3.1 Käyttöintensiteetti

Järjestelmä on yhden käyttäjän järjestelmä, mutta laitteiston tulee kuitenkin kyetä toimimaan yhtäjaksoisesti.

Laitteiston käyttöintensiteetti saattaa tuotantotestauksessa olla jopa 24 tuntia vuorokaudessa, jolloin se on suunniteltu kestämään ympärivuorokautista käyttöä.

3.2 Kapasiteettivaatimukset

Laitteiston tiedontallennustila riippuu käytettävästä CF-kortista.

Laitteiston tulee kyetä mittaamaan mittausarvot enintään 10 kertaa sekunnissa. Mittaukset eri kanavista tehdään vuorotellen.

Tiedontallennus-kapasiteetti ja ajallinen pituus riippuvat mittausnopeudesta ja käytettävän muistin koosta.

3.3 Tiedostot ja asetustiedostot

Laitteiston asetustiedot löytyvät mikro-ohjaimen sisäisestä EEPROM muistista (ks. muistikartta).

Mittaustiedon tallennus tehdään muistikortille FAT16 –tiedostojärjestelmään johon luodaan mittaus-data –tiedosto ASCII muotoisena tekstitiedostona, jonka sarake-erottimena toimii ”;” –merkki.

Tiedosto nimetään oletus-arvoisesti data_accX.txt nimellä, jossa X on juokseva numero, käyttäjä voi nimetä tiedoston ennen testin alkua.

Ohjelmisto ei sisällä muita tiedostonhallinta-ominaisuuksia

4. TOIMINNOT

4.1 Käynnistys

Laite käynnistetään kytkemällä siihen sähköt laitteen pääkytkimestä, jonka jälkeen sähköt kytkeytyvät koko laitteistoon, paitsi testattavaan ovioperaattori-korttiin, riippuen erillisen kytkimen asennosta.

Kun laite käynnistyy, se alustaa ja lataa tallennetut parametrit muistista automaattisesti. Tämän jälkeen laite tarkistaa CF-muistikortin tilan ja käytössä olevan tilan.

Tämän jälkeen käyttäjälle näytetään laitteistotiedot ja se jää valmius-tilaan, jossa sen asetuksia voidaan tarkistaa ja muuttaa käyttöliittymästä tai sarjaliikenne-ohjelmiston kautta.

4.2 Järjestelmän toiminnot

4.2.1 Vastavoimamoottorin ohjaus ja takaisinkytkentä

Vastavoimamoottoria ohjataan kaupallisen taajuusmuuttajan avulla, jota ohjataan DLMCB –kortilla. Ohjattava taajuusmuuttaja on Vacon NXP00035A2 –tyyppinen.

Ohjainkortti syöttää toimilaitteelle, eli taajuusmuuttajalle virta-viestinä momenttiohjetta (4-20mA) ja useita eri digitaalisia (ON/OFF) ohjeita, joiden perusteella taajuusmuuttaja ohjaa moottoria.

Virtaviesti muodostetaan ulkoisella D/A –muunninpiirillä AD7524. Virtaviesti syötetään D/A-muuntomelle 8bittisesti I/O-data –väylää pitkin, kun piiri on ensin aktivoitu I/O-valitsimella (ks luku 3.1.6).

Taajuusmuuttaja antaa takaisinkytkentäsignaalin moottorin momentista analogia-kanavaan virtaviestinä (4-20mA) josta se muutetaan mikro-ohjaimen A/D-muuntimelle sopivaksi jänniteviestiksi (0-5)- Takaisinkytkentätieto voidaan lukea A/D-muunnin kanavasta 7 (ks luku 3.1.3).

Takaisinkytkentätieto on valinnainen tallennustieto.

4.2.2 Mitattavat tiedot

Laite mittaa käyttäjän asetusten mukaisesti vuorotellen AMD-ovioperaattorin moottorille syöttämät jännitteet ja virrat. Riippuen

parametroinnista mitattavia jännitteitä/virtoja on kaksi (AMD1) tai kolme (AMD2).

Lisäksi mikro-ohjain mittaa ulkoisen taajuusmuuttajan takaisinkytkentäsignaalia.

4.2.3 Tiedon tallennus

Tiedot tallentaan CF-muistikortille ASCII tiedostoksi siten että sarake-erottimena toimii ”;”-merkki.

Tallennettavat tiedot käyttäjä voi valita asetuksista itse.

Tallennettavat tiedot ovat aina muotoa:

time	U1	U2	U3	I1	I2	I3	FB

Jotka esitetään data-tiedostossa muodossa:

time;U1;U2;U3;I1;I2;I3;FB;

Tiedoston alkuun kirjoitetaan myös tiedoston nimi ja mittauksen aloitus-aika.

time –muuttuja tiedostossa kertoo kuluneen ajan aloitushetkestä.

4.2.4 Ovi-operaattorin ohjaus

Ovi-operaattorin ohjaus ”stand-alone” –tilassa tapahtuu kuten se tapahtuisi reaalisessakin kohteessa, täysin hissin ovi-ohjaus-standardin mukaisesti.

Laitteella voidaan simuloida myös oven kaikki turvalaitteet.

Lisätiedot hissin ovi-operaattorin ohjauksesta löytyvät KONE-dokumentista.

AMD1: AM-03.35.022

AMD2: AM-03.35.020 ja AM-03.35.023

4.2.5 Asetukset

Ennen kortin testausta, riippuen laitteiston versiosta, tulee jotkin parametrit asettaa kalustusversion mukaisiksi. Nämä parametrit löytyvät parametriluettelosta (ks liite 2).

Muut asetukset ovat valinnaisia (ks liite 2).

4.2.6 Virhe ja varoitus ilmoitukset ja toiminnot

Virhe-toiminnan sattuessa laitteisto pysäyttää kuormamoottorin toiminnan välittömästi.

Lisäksi, jos käytössä on ”stand-alone”-tila, laitteisto lopettaa AMD-ovioperaattorin ohjauksen siten että sen tulee pysäyttää moottori. Tämä tehdään kuitenki vasta kun, kuormamoottorin ohjaus on pysäytetty, eli se pyörii vapaasti.

Pysäytyksen jälkeen laitteisto antaa käyttäjälle virheilmoituksen käyttöliittymänäytöllä.

Varoituksen aiheuttava toiminto ei pysäytä testiä, vain varoitusilmoitus näytetään käyttäjälle ruudulla. Tälläinen toiminta on esimerkiksi muisti-tilan loppuminen kesken testauksen.

Lisätietoja vioista ja varoituksista on annettu liittessä 3, vika- ja varoituslista.

5. ULKOISET LIITTYMÄT

5.1 Laitteistoliittymät

Laitteeseen voidaan kytkeä CF-muistikortti tiedon tallennusta varten.

Laitteistoon voidaan myös ottaa yhteys RS-232 sarjaliikenneväylän kautta.

Muutoin laitteistoliittymänä toimivat 2x20 LCD-näyttö ja käyttöliittymänäppäimet.

5.2 Ohjelmistoliittymät

Ohjelmistoon voidaan ottaa yhteyttä standardi-sarjaliikennesovelluksella standardin mukaista RS-232 sarjaliikenneväylää käyttäen.

5.3 Tietoliikenneliittymät

Laitteisto käyttää standardin mukaista RS-232 sarjaliikenneväylää.

9600bps, 8 databit, 1 stop, Flowcontrol: None.

6. MUUT OMINAISUUDET

Tässä luvussa kuvataan ne ei-toiminnalliset ominaisuudet, jotka eivät ole tulleet esille aikaisemmissa luvuissa.

6.1 Suorituskyky ja vasteajat

Järjestelmän tulee kyetä mittaamaan kaikki mittaus-kohteet enintään 10 kertaa sekunnissa vuorotellen ja tallentaa tieto jos asetus on valittu ja muistitilaa on vapaana.

6.2 Käytettävyys, toipuminen, turvallisuus, suojaukset

Laitteen käyttöliittymä toimii neljällä näppäimellä, näppäimillä on myös muita toimintoja riippuen toimintatilasta.

Turvallisuus ja suojaustoiminnot on esitetty luvussa 4.2.6.

Ohjain-kortilla on aktivoitu watchdog –ajastin, joka käynnistää prosessorin uudelleen, jos ohjelmisto jumiutuu.

6.3 Ylläpidettävyys

Järjestelmää ylläpitää Jani Angervuo. Ohjelmakoodiin tehtävät korjaukset tehdään suoraan versiohallinnasta löytyvään lähdekoodiin.

6.4 Siirrettävyys ja yhteensopivuus

Laitteisto on yhteensopiva minkä tahansa standardin mukaisen RS-232 sarjaliikenneväylä-päätteen kanssa.

Laite on yhteensopiva AMD1 ja AMD2 ohjainkorttien kanssa.

6.5 Operointi

Järjestelmän operointi ei vaadi käyttäjältä erityisosaamista. Käyttäjän tulee kuitenkin saada käyttökoulutus.

7. SUUNNITTELURAJOITTEET

7.1 Standardit

Laitteisto käyttää CF-muistikorttia, jonka toiminnan tulee tukea trueIDE –mode –toimintaa.

Laitteistossa käytetään FAT16/32 –tiedostomuotoisen tiedoston kirjoittamiseen ja tallentamiseen ” FAT16-32 File System Driver for ATMEL AVR - V1.01” –tiedostokirjastoa, joka on GNU-lisenssin alainen ajurikirjasto.

Kirjastoa on muokattu siten että se sopii käytettävälle kääntäjälle.

Laitteisto tukee Electronic Industries Alliancen suosituksen mukaista EIA232 sarjaliikenneprotokollaa.

7.2 Laitteistorajoitteet

Ohjelmoitava Flash –muisti	64kB
EEPROM	2kB
Mikro-ohjaimen nopeus	16 MIPS
Tallennus-kapasiteetti	CF-kortti

7.3 Ohjelmistorajoitteet

Tallennustiedostot kirjoitetaan CF-muistikortille FAT16/32 muotoon käytettävän ohjain-kirjaston rajoitusten mukaisesti. Kirjaston tiedot:

```
// Title        : FAT16/32 file system driver for ATMEL AVR
// Authors      : Angelo Bannack, Giordano Bruno Wolaniuk
// Date                 : April 26, 2004
// Version      : 1.00
// Target MCU   : Atmel AVR Series
```

Kirjoitettava tiedosto kirjoitetaan ASCII tekstitiedostoksi luvussa 3.4. esitetyllä tavalla.

Ohjelmointiympäristönä ja kääntäjänä:

CodevisionAVR Standard. Käytettävä versio vähintään 1.23.8c